PRACTICAL MICROPROCESSORS

edizione in lingua italiana

HEWLETT PACKARD

hardware, software e ricerca guasti

I MICROPROCESSORI IN PRATICA

hardware, software e ricerca guasti

Michael Slater

Barry Bronson

JACKSON ITALIANA EDITRICE
Piazzale Massari, 22
20125 Milano

© 1979 per l'edizione originale Hewlett-Packard Company,
5301 Stevens Creek Blvd. Santa Clara, Calif. 95050

© 1980 Hewlett-Packard Italiana

Edizione a cura della Jackson Italiana Editrice, Milano

Prima edizione 1980

Collaborazione tecnico-editoriale a cura dell'I.D.E.A. Studio s.r.l., Milano

Stampato in Italia da FOR-VEM, Milano

INDICE DEGLI ARGOMENTI

INTRODUZIONE ALL'EDIZIONE ITALIANA xix

PREFAZIONE xxi

I. NOZIONI FONDAMENTALI SUI MICROPROCESSORI 1

Lezione 1: Introduzione ai sistemi a microprocessore 3
- Introduzione 3
- Lo sviluppo del microprocessore 3
- Perché vengono usati i microprocessori 5
- Un sistema base con microprocessore 6
- I programmi 8
- Periferiche 8
- Dispositivi a tre stati 9
- Il microprocessore 11
- Memorie 12
- RAM e ROM 13
- Microcalcolatori e minicalcolatori 16
- Esperimento 1-1: Introduzione al Microprocessor Lab 17
- Ripasso 19
- Domande 20

Lezione 2: Sistemi di numerazione 21
- Introduzione 21
- Decimali e binari 21
- Ottali 22
- Esadecimali 23
- Nomenclatura dei bit 24
- Ripasso 25
- Domande 26

Lezione 3: Elementi di software 27
- Introduzione 27
- I calcolatori non pensano 27
- Il microprocessore nelle vesti di un dispositivo logico 28
- Diagrammi di flusso 29
- Esperimento 3-1: Il dispositivo AND col Microprocessor Lab 31
- Caratteristiche del dispositivo AND realizzato con un microprocessore 32
- Linguaggi di programmazione 32
- Un esempio di programmazione 33
- Confronto tra i diversi tipi di linguaggi 37
- La programmazione del μLab 38
- Esperimento 3-2: Interpretazione del contenuto della memoria 39
- Un esempio di applicazione 41
- Modificare un programma 42
- Esperimento 3-3: Simulazione del nastro trasportatore 44
- Ripasso 45
- Domande 46

INDICE DEGLI ARGOMENTI

(continuazione)

II. INTRODUZIONE ALLA PROGRAMMAZIONE 47

Lezione 4: Uso del Microprocessor Lab 49
Introduzione 49
La mappa di memoria del Microprocessor Lab 49
Esperimento 4-1: Esame della memoria e memorizzazione dei dati 51
Un programma semplice 54
Esperimento 4-2: Esecuzione dei programmi 55
Le porte di ingresso e d'uscita 59
Esperimento 4-3: Uso delle porte di ingresso e d'uscita 61
Ripasso 63
Domande 64

Lezione 5: Alcuni concetti di software 65
Introduzione 65
I registri del microprocessore 65
Seguire il flusso di un programma 66
Il programma monitor del μLab 66
Un programma contatore 67
Esperimento 5-1: Esecuzione del programma contatore 69
Organizzazione del programma 73
Un esempio 74
Esperimento 5-2: Subroutine 76
Interruzioni 77
L'uso del tasto interrupt del μLab 77
Esperimento 5-3: Interruzioni 80
Ripasso 81
Domande 82

Lezione 6: All'interno del microprocessore 83
Introduzione 83
All'interno del microprocessore 8085A 83
Il ciclo d'istruzione 84
Esecuzione delle istruzioni 85
Cicli macchina 86
Esecuzione di un programma 86
Esperimento 6-1: Come opera il bus 86
Ripasso 91
Domande 92

III. HARDWARE DEL SISTEMA A MICROPROCESSORE 93

Lezione 7: Concetti fondamentali di hardware 95
Introduzione 95
Il concetto di bus 95
Il bus a tre stati 96
Il bus dei dati 98
Il bus degli indirizzi 99

(continuazione)

Decoder degli indirizzi 100
Il bus di controllo 100
Porte d'uscita 101
Porte di ingresso 101
Decodifica degli indirizzi per più dispositivi 101
Decodifica degli indirizzi per le memorie 102
Controllo di più dispositivi di memoria 102
Controllo della RAM 104
Ripasso 106
Domande 107

Lezione 8: Decodifica degli indirizzi 109
Introduzione 109
Strutture di indirizzamento 109
La struttura di indirizzamento del Microprocessor Lab 109
L'hardware di decodifica 111
Il circuito di protezione della RAM 112
Esperimento 8-1: Il decoder degli indirizzi 114
Altre tecniche di decodifica 119
Decodifica con selezione lineare 119
Decodifica con comparatore logico 119
Decodifica con logica combinatoria 120
Decodifica con mappa in I/O 121
Ripasso 122
Domande 123

Lezione 9: Memorie e periferiche 125
Introduzione 125
Memorie 125
RAM 125
La RAM del Microprocessor Lab 126
Altre configurazioni di memoria 126
Memorie a sola lettura (ROM) 127
La ROM del Microprocessor Lab 128
Periferiche del microcalcolatore 128
Ingressi ed uscite 128
Tastiera e display 129
Le porte d'uscita seriale 132
Esperimento 9-1: Esecuzione dei programmi 133
Chip di interfaccia a periferiche 139
Ripasso 141
Domande 142

Lezione 10: Circuiti di controllo 145
Introduzione 145
La porta di controllo 145
Il bus in multiplex 146
La famiglia 8085 148

INDICE DEGLI ARGOMENTI

(continuazione)

Esperimento 10-1: Temporizzazioni del bus 149
Altri segnali di controllo 153
Il circuito di single-step 156
Timer programmabili 157
Considerazioni elettroniche 157
Ripasso 162
Domande 163

IV. SOFTWARE DEI MICROPROCESSORI 165

Lezione 11: Registri e breakpoint 167
Introduzione 167
Istruzioni già descritte 167
Perché si hanno parecchi tipi di istruzioni 168
I registri di utilizzo generale 169
Indirizzamento indiretto 169
Esperimento 11-1: Utilizzazione dei registri 171
Breakpoint 174
Utilizzo dei breakpoint 174
Breakpoint hardware 174
Esperimento 11-2: Uso dei breakpoint 175
Ripasso 177
Domande 178

Lezione 12: Istruzioni dell'8085 179
Introduzione 179
Istruzioni logiche 179
Esperimento 12-1: Istruzioni logiche 181
Mascheratura 183
Esercizio di programmazione N. 1: Mascheratura 183
Azzeramento dell'accumulatore 184
Operazioni di shift 184
Esercizio di programmazione N. 2: Rotazioni 185
Addizione 186
Flag di riporto 186
Sottrazione 186
Esperimento 12-2: Funzioni aritmetiche 188
Subroutine e stack 190
Istruzioni Push e Pop 191
Ripasso 193
Domande 194

Lezione 13: Tecniche di progettazione software 195
Introduzione 195
Sistemi di sviluppo 195
Come si procede in un progetto software 196
Esempio di progetto software 197
La routine sequenza 198

La routine cambio 198
Routine di ritardo 200
Schemi di struttura 201
Programmi di controllo 201
Routine di ritardo 203
Utilizzo di coppie di registri 203
Tecniche di collaudo 205
Esperimento 13-1: Collaudo del programma di controllo per semafori 208
Come migliorare il controllo per semafori 210
Esperimento 13-2: Modifica del controllo per semafori 211
Esercizio di programmazione N. 3: Modifica del controllo semafori 212
Ripasso 213
Domande 214

Lezione 14: Controllo software delle periferiche 215
Introduzione 215
Tastiera 215
Esperimento 14-1: Utilizzazione della routine di lettura tastiera 216
Esercizio di programmazione N. 4: Antifurto elettronico 218
Scansione della tastiera 218
Esperimento 14-2: Scansione della tastiera 220
Antirimbalzo 222
Il display 222
Esperimento 14-3: Visualizzazione di un messaggio 224
Esercizio di programmazione N. 5: Uso della tastiera e del display 227
Controllo diretto del display 227
Esperimento 14-4: Controllo diretto di un display 228
Scansione di tutte le cifre 230
Esperimento 14-5: Scansione del display 232
Ripasso 233
Domande 234

Lezione 15: Rappresentazioni numeriche e algoritmi 235
Introduzione 235
Numeri negativi 235
Piccoli e grandi numeri 236
Rappresentazione dei numeri decimali 238
Rappresentazione dei caratteri alfanumerici 238
Table Look-Up 239
Algoritmi matematici 240
Ripasso 242
Domande 243

V. RICERCA DEI GUASTI NEI SISTEMI A MICROPROCESSORE 245

Lezione 16: Strumenti portatili per la ricerca dei guasti 247
Introduzione 247
Esperimento 16-1: Sonde logiche 248
Esperimento 16-2: Il generatore di impulsi logici 255

INDICE DEGLI ARGOMENTI

(continuazione)

Esperimento 16-3: Il collaudo stimolo-risposta con una sonda e un generatore di impulsi 259
Esperimento 16-4: Il rivelatore di corrente 263
Ripasso 269
Domande 270

Lezione 17: Analizzatori di firma e analizzatori logici 271
Introduzione 271
Analisi di firma 271
Tabelle di firma per il Microprocessor Lab 272
Esperimento 17-1: Funzione free-run del Microprocessor Lab 274
Esperimento 17-2: Prova della ROM durante il free-run 280
Esperimento 17-3: Loop di test per SA 282
Analizzatori logici 288
Ripasso 292
Domande 293

Lezione 18: Ricerca di guasti nei sistemi a microprocessore 295
Introduzione 295
Problemi di ricerca guasti nei microprocessori 295
Problemi specifici dei sistemi a microprocessore 297
Programmi di autodiagnostica 299
I/O in multiplex 300
Interfacce 300
Alberi di ricerca guasti 300
Altra documentazione 301
Esiste realmente un problema? 302
Che cosa si può ricavare dal pannello frontale? 302
Che cosa dice il manuale? 302
Guasti in produzione e sul campo 303
Quali sono le cose più semplici da provare? 304
Tipici problemi di ricerca guasti in produzione 304
Guasti meccanici in prodotti già installati 305
Tecniche generali di ricerca guasti 305
Come può essere isolato il guasto? 308
Tipi di guasti digitali 308
Tecniche di isolamento 310
Anelli di reazione 310
Anelli di reazione 310
Conclusioni 312
Ripasso 313
Domande 314

Lezione 19: Ricerca di guasti nel Microprocessor Lab 315
Introduzione 315
Il diagramma di flusso per la ricerca guasti del Microprocessor Lab 315
Esperimento 19-1: Familiarizzazione con la procedura di ricerca guasti 319

Ricerca dei guasti con un oscilloscopio 327
Alla ricerca dei guasti 327
Caratteristiche atipiche del μLab 329
Ripasso 330
Domande 331

VI. UNA PANORAMICA DEGLI ALTRI MICROPROCESSORI 333

Lezione 20: Panoramica dei microprocessori 335
L'8085 e gli altri microprocessori 335
I microprocessori a quattro bit 335
I microprocessori a sedici bit 336
Microcalcolatori su un solo chip 336
I processori bit-slice 336
Descrizione dei microprocessori 337
Ripasso 343
Domande 344

APPENDICI

A. Soluzioni dei problemi 347
Risposte alle domande 348
Soluzioni agli esercizi di programmazione 349
Programmi per esperimenti 352
Soluzioni dei problemi di ricerca guasti 353

B. Le istruzioni dell'8085A 365
Introduzione 365
Formato delle istruzioni e dei dati 368
Modi di indirizzamento 368
Flag di condizione 369
Simboli ed abbreviazioni 369
Descrizione del formato 370
Gruppo trasferimento dati 371
Gruppo aritmetico 375
Gruppo logico 381
Gruppo di salto 386
Gruppo stack, I/O, controllo macchina 390

C. Tabelle delle firme 395

D. Lettura degli schemi elettrici 405

E. Programmi dimostrativi e di utilità 407
Programmi dimostrativi 407
Programmi di utilità 410

F. Listing della ROM del Microprocessor Lab 413

G. Espansione del Microprocessor Lab 423

H. Fogli tecnici dei circuiti integrati 429

INDICE DEGLI ARGOMENTI

(continuazione)

GLOSSARIO 437

BIBLIOGRAFIA 449

DIAGRAMMI HARDWARE DI RIFERIMENTO 451

RIASSUNTO DELLE FUNZIONI . . . . . Risvolto di copertina

INDICE DELLE FIGURE

Figura	Titolo	Pagina
1-1	Sistema base a microprocessore	7
1-2	Voltmetro digitale basato su microprocessore	8
1-3	Il driver a tre stati	9
1-3	Equivalente funzionale del driver	9
1-5	Schema di una tipica uscita a tre stati	9
1-6	Attraverso dei driver a tre stati è possibile portare su una sola linea di dato più segnali	10
1-7	Driver a tre stati in un sistema a microprocessore	10
1-8	I segnali principali di un microprocessore	11
1-9	Diagramma di una memoria da otto bit in cui è mostrato come tutte le celle di memoria siano collegate alla stessa linea di dato	13
1-10	Una ROM da 2K ×8	15
1-11	Una RAM da 1K×8	15
1-12	Come preparare la valigetta del Microprocessor Lab	17
1-13	Selezione della tensione di rete	17
2-1	Sistemi di numerazione binaria e decimale	21
2-2	Conversione da decimale a binario	23
2-3	Rappresentazione ottale dei numeri	23
2-4	Rappresentazione esadecimale (Hex) dei numeri	23
3-1	Livelli di intelligenza	27
3-2	Dispositivo AND realizzato con un microprocessore	28
3-3	Simboli dei diagrammi di flusso	29
3-4	Diagramma di flusso del dispositivo AND	29
3-5	Diagramma di flusso del programma «Conta a 10»	33
3-6	Sistema con nastro trasportatore	41
3-7	Diagramma di flusso del programma di controllo	42
3-8	Diagramma di flusso del controllore modificato	43
4-1	Mappa di memoria del Microprocessor Lab	49
4-2	Diagramma di flusso del programma «Fannullone»	54
4-3	Diagramma di flusso del programma che copia i dati dalla porta d'ingresso su quella d'uscita	59
5-1	Registri e Memorie	65
5-2	Programma che fa sì che la porta d'uscita conti in binario	67
5-3	Utilizzo di subroutine per far lampeggiare i LED della porta d'uscita	73
5-4	Sequenza semplificata degli eventi che si verificano quando il programma contatore viene interrotto	78
5-5	Flusso dettagliato del programma quando il programma contatore viene interrotto	79
6-1	Schema a blocchi semplificato dell'8085	83
6-2	Lettura del codice operativo, dalla memoria, dell'istruzione MVI A	84
6-3	Lettura del dato dell'istruzione MVI A	85

INDICE DELLE FIGURE

(continuazione)

Figura	Titolo	Pagina
7-1	Trasferimenti di dati secondo tecniche di progetto tradizionali	95
7-2	Trasferimento dati attraverso un bus in modo da ridurre il numero dei collegamenti	96
7-3	Bus a tre stati, formato da una sola linea, con quattro dispositivi «parlatori» e due «ascoltatori»	96
7-4	Logica di controllo che seleziona i dispositivi che sono interessati al trasferimento dei dati	97
7-5	Parlatori-ascoltatori bidirezionali collegati alla linea del bus	97
7-6	Dispositivi con uscita a tre stati che comunicano con il microprocessore attraverso il bus dei dati	98
7-7	Decoder degli indirizzi che controlla la porta di indirizzo 3000	99
7-8	Il dato presente nel bus dei dati è immagazzinato in un latch, ogni volta che il microprocessore scrive nell'indirizzo 3000	100
7-9	I dati d'ingresso sono posti sul bus dei dati ogni volta che il microprocessore legge l'indirizzo assegnato al driver a tre stati	101
7-10	Un IC decoder permette di estendere facilmente il numero dei dispositivi che possono essere selezionati	101
7-11	Il decoder interno alla ROM permette di ridurre il numero di linee di indirizzo richieste ad un decoder esterno	103
7-12a e b	Indirizzi assegnati a ciascuna delle quattro ROM da 256 byte presenti in un sistema	103
7-13	Decodifica degli indirizzi per le quattro ROM da 256 byte dell'esempio in Figura 7-12	104
7-14	Tabella della verità per il controllo della RAM	104
7-15	Controllo e decodifica degli indirizzi per una RAM da 1K byte facendo uso della tabella della verità della Fig. 7-14	105
7-16	Schema a blocchi del μLab	105
7-17	Circuito di decodifica degli indirizzi cui fanno riferimento le domande 6, 7 e 8	108
8-1	Mappa completa degli indirizzi per il μLab	110
8-2	Circuito di decodifica degli indirizzi nel μLab	111
8-3	La RAM è abilitata (segnale basso) ogni volta che in questo loop di programma viene letto un byte di istruzione	115
da 8-4 a 8-9	L'ordine secondo cui i dispositivi vengono selezionati segue l'ordine delle istruzioni del programma	116
8-10	Decodifica con selezione lineare	119
8-11	Decodifica con un comparatore logico	120
8-12	Decodifica con un dispositivo logico NAND	120
8-13	La decodifica con mappa di I/O aumenta di 256 byte l'area indirizzabile	121
9-1	Circuito RAM del μLab	126
9-2	Memoria da 1K $\times$ 8 realizzata con RAM 2102 (1K $\times$ 1)	127
9-3	Circuito ROM del μLab	128
9-4a	Porta di ingresso del μLab	128
9-4b	Porta d'uscita del μLab (continuazione)	129
9-5	Interfaccia tastiera simile a quella del μLab	130

Figura	Titolo	Pagina
9-6	Dati di scansione quando è premuto il tasto indicato nella Figura 9-5	130
9-7	Interfaccia display simile a quella del μLab	131
9-8	Circuito di uscita seriale del μLab	132
9-9	Segnali di selezione chip mentre è in esecuzione un breve programma a loop	134
9-10	Segnale di selezione ROM mentre sta girando il programma monitor	135
9-11	Segnale di selezione RAM mentre sta girando il programma monitor	135
9-12	Segnale di selezione KYRD mentre sta girando il programma monitor	136
9-13	Segnale di selezione KYRD mentre sta girando il programma monitor	137
9-14	Segnale di selezione SCAN mentre sta girando il programma monitor	137
9-15	Segnale di selezione DSP mentre sta girando il programma monitor	138
9-16	Dispositivo d'interfaccia periferiche contenente tre porte di I/O da otto bit	139
9-17	Gli UART realizzano l'interfaccia di comunicazione seriale tra due sistemi	140
10-1	Porta di controllo del μLab	145
10-2	Circuito di demultiplex degli indirizzi utilizzato nel μLab	146
10-3	Temporizzazioni del sistema 8085	147
10-4	I fronti di discesa di ALE indicano che su AD0 sono presenti degli indirizzi stabili	149
10-5	La linea di indirizzo A0 dopo il demultiplex, generata a partire dalla linea di indirizzo in multiplex ADO	150
10-6	Il segnale ALE controllo A0	151
10-7	I dati stabili presenti sulla linea AD0 sono letti dal microprocessore sul fronte di salita del segnale $\overline{\text{READ}}$	151
10-8	Quando il segnale $\overline{\text{READ}}$ è alto e ALE è basso, la linea AD0 non contiene segnali logici significativi	152
10-9	Circuiteria d'interruzione del μLab	153
10-10	Il circuito di single-step fa sì che il microprocessore proceda di un solo ciclo macchina	156
10-11	Buffer sul bus dei dati del μLab	158
10-12	I dati sono caricati in memoria sul fronte di salita del segnale $\overline{\text{WRITE}}$	159
10-13	I dati vengono letti dal microprocessore sul fronte di salita del segnale $\overline{\text{READ}}$	159
10-14	Ciclo d'istruzione tipico	160
10-15	Temporizzazioni delle fasi di prelievo e di esecuzione dell'istruzione OUT	161
11-1	Schema a blocchi semplificato dell'8085 in cui sono evidenziati i registri di utilizzo generale	168
11-2	Indirizzamento indiretto attraverso i registri H e L	169
12-1	Equivalente hardware dell'istruzione AND	180
12-2	Diagramma di flusso del programma che effettua il test del bit 3 della porta di ingresso	183
12-3	Come operano le istruzioni di rotazione	185
12-4	Diagramma di flusso per l'esercizio di programmazione N. 2	185
12-5	I flag del microprocessore	187

INDICE DELLE FIGURE

(continuazione)

Figura	Titolo	Pagina
12-6	Sequenza di eventi relativi al programma principale che chiama la routine A che chiama a sua volta la routine B	191
12-7	Come opera lo stack	191
13-1a	Programma principale per il controllo dei semafori	198
13-1b	Routine seguenza	198
13-2	Collegamenti dei LED per simulare i semafori	199
13-3	Routine di cambio semaforo (CHNG)	200
13-4	Routine di ritardo	200
13-5	Schema funzionale del controllore di semafori	201
13-6	Routine per un ritardo lungo	204
14-1	Interfaccia tastiera	218
14-2	Rimbalzo di un interruttore	222
14-3	Diagramma di flusso del programma antirimbalzo	223
14-4	Controllo del display del μLab	227
15-1	Doppia precisione	237
15-2	Virgola fissa	237
15-3	Virgola mobile	237
15-4	Programma con accesso in tabella per la conversione da binario a sette segmenti	240
15-5	Moltiplicazione binaria	241
16-1	Collegamento dell'alimentazione della sonda effettuato attraverso uno zoccolo per il test dei circuiti integrati	248
16-2	Come si usano gli «uncini» per collegare la sonda ai fori di alimentazione del μLab	249
16-3	Soglie di tensione TTL per la sonda logica 545A	250
16-4	Interruttori della porta d'ingresso del μLab	250
16-5	Circuito di scansione e di lettura tasti relativo alla colonna dei tasti 0-D	251
16-6	Circuito di controllo del single-step	253
16-7	Forma d'onda che esce da un generatore di impulsi logici 546A nel caso di un nodo normalmente basso	255
16-8	Circuito di pilotaggio dell'altoparlante del μLab	256
16-9	Circuito latch della porta d'uscita	257
16-10	Sonda logica inserita nel foro di prova D0	259
16-11	Circuito di decodifica della RAM	260
16-12	Generatore di impulsi logici utilizzato per incrementare il circuito single-step	261
16-13	Orientazione corretta del rivelatore di corrente 547A rispetto alla traccia del circuito	263
16-14	Attività della corrente per diversi circuiti del μLab	265
16-15	Misura con un rivelatore di corrente della corrente iniettata dal generatore d'impulsi in un nodo	266
16-16	Come seguire la corrente sul retro della scheda servendosi del rivelatore di corrente	267

Figura	Titolo	Pagina
17-1	Analizzatore di firma 5004A	272
17-2	Posizione degli interruttori di test del μLab	274
17-3	Circuito utilizzato per aprire il bus dei dati e far operare il μLab in free-run	275
17-4	Come collegare al μLab l'analizzatore di firma nel caso del test degli indirizzi in free-run	
17-5	Nel corso del test degli indirizzi in free-run i dati vengono campionati entro finestre di misura di 64K indirizzi	277
17-6	Circuito di pilotaggio e demultiplex degli indirizzi	278
17-7	Circuito di decodifica degli indirizzi	279
17-8	Nel corso del test della ROM in free-run i dati vengono campionati solo durante i 2K indirizzi in cui viene letta la ROM	280
17-9	Verifica dei LED d'uscita durante il loop di test SA	283
17-10	Circuito di ingresso tastiera	285
17-11	Posizione del ponticello W 10 di guasto	286
17-12	Circuito della porta d'uscita	287
17-13a	L'analizzatore logico è utilizzato per visualizzare indirizzi e dati in forma tabellare ed è così possibile seguire l'esecuzione del programma	289
17-13b	Le stesse informazioni della Figura 17-13a ma presentate in forma binaria	289
17-14	Le otto linee del bus dei dati visualizzate nel modo «analisi temporale»	290
17-15	L'analizzatore logico viene utilizzato nel modo «a mappa espansa» al fine di mostrare l'attività degli indirizzi	291
17-16	La mappa visualizzata durante il modo free-run indica una uguale attività per tutti i 64K indirizzi	291
18-1	Albero di ricerca guasti tipico di un prodotto che preveda l'analisi di firma	301
18-2	Uso di un voltmetro ad alta sensibilità per localizzare dei corti sulle linee di alimentazione	307
18-3	I conflitti sul bus determinano scorretti, ma ben definiti livelli logici	309
18-4	La rottura del diodo di protezione dell'ingresso, posto sul chip, determina un cortocircuito tra un piedino di ingresso e una massa	309
19-1	Diagramma per la ricerca dei guasti nel Microprocessor Lab	316
19-2	Posizione del ponticello W 1 di guasto e fori di test per la sonda sul bus degli indirizzi	319
19-3	Come preparare l'analisi di firma per il test degli indirizzi in free-run	322
19-4	Impulsi di corrente sulla linea A11 degli indirizzi	324
19-5	Regolazione della sensibilità del rivelatore di corrente	324
19-6	Punto dove cambia la corrente	325
19-7	Schermo dell'oscilloscopio con A10 e A11 in corto	327
19-8	Tutti e dodici i ponticelli di guasto hanno due possibili posizioni: normale e guasto	328
20-1	CPU in tre chip basata sull'8080	337
20-2	Segnali di controllo del 6800: operazioni di lettura	338
20-3	6800: operazione di scrittura	339
20-4	Registri della CPU 6800	340
20-5	Schema a blocchi semplificato del sistema F8	342

INDICE DELLE FIGURE

(continuazione)

Figura	Titolo	Pagina
B-1	Riferimenti per il linguaggio di assemblaggio 8085/8080	366
D-1	Schemi logici funzionali	405
G-1	Un circuito che permette di operare in single-step pur utilizzando un ingresso di READY esterno	427
	Schema piste della scheda 5036A	452
	Schema elettrico del Microprocessor Lab 5036A	453
	Schema a blocchi del Microprocessor Lab 5036A	454

INDICE DELLE TABELLE

Tabella	Titolo	Pagina
2-1	Sistemi di numerazione	22
3-1	Programma «Conta a 10» in linguaggio BASIC	34
3-2	Programma «Conta a 10» nel linguaggio d'assemblaggio dell'8085	34
3-3	Listing in linguaggio macchina dell'8085 per il programma «Conta a 10»	36
3-4	Programma «Conta a 10» in tre linguaggi	36
3-5	Elenco delle istruzioni per l'esperimento N. 3	39
4-1	Listing del programma	54
4-2	Programma che copia i dati dalla porta d'ingresso alla porta d'uscita	59
4-3	Programma modificato che complementa il dato	62
5-1	Listing del programma contatore	68
5-2	Programma che fa lampeggiare i LED utilizzando delle subroutine	74
5-3	Routine di abilitazione dell'interruzione	77
5-4	Routine di servizio dell'interruzione	78
6-1	Tabella del programma dell'esperimento 6-1	89
11-1	Programma dimostrativo sui registri	171
11-2	Registri dell'8085	172
11-3	Programma contatore con breakpoint	175
12-1	Programma per spiegare le istruzioni logiche	181
12-2	Programma test bit 3	184
12-3	Esempio di programma aritmetico	188
13-1	Configurazioni dei bit per simulare i semafori	199
13-2	Programma principale per il controllo semafori (TRAF)	201
13-3	Routine sequenza semaforo (SEQ)	202
13-4	Routine cambio semaforo (CHNG)	202
13-5	Semplice routine di ritardo (DELAY)	203
13-6	Routine di ritardo che fa uso di una coppia di registri	203
13-7	Routine di ritardo per il controllo di un semaforo	205
13-8	Programma per il controllore dei semafori	206
13-8	Programma per il controllore dei semafori (continuazione)	207
13-9	Programma principale per il controllo dei semafori con diverse durate del verde	210
14-1	Codici dei tasti per la routine KIND	215
14-2	Programma per leggere la tastiera e generare un bip se viene premuto il tasto «7»	216
14-3	Codici tasti	219
14-4	Programma che legge una riga dalla tastiera	219
14-5	Programma che verifica quando è premuto il tasto «2»	220
14-6	Programma per visualizzare un messaggio	224

INDICE DELLE TABELLE

(continuazione)

Tabella	Titolo	Pagina
14-7	Codici dei caratteri per la routine DCD	225
14-8	Programma per visualizzare un «2»	228
14-9	Subroutine display	230
14-10	Programma di scansione display	231
15-1	Rappresentazione in complemento a due dei numeri da −8 a +7	235
15-2	Codici ASCII	239
B-1	Riferimenti per il linguaggio di assemblaggio 8085/8080	366
C-1	Firme in free-run relative agli indirizzi	396
C-2	Firme in free-run relative alla ROM	398
C-3	Firme in S.A. di scrittura	400
C-4	Firme in S.A. di lettura	402
D-1	Relazioni tra i dispositivi (gate) fondamentali	406
E-1	Programmi dimostrativi	407
E-2	Note musicali per il programma Organ	408
E-3	Quando rispondere	409
E-4	Routine di utilità	410
E-5	Cifre decodificate del display	411
E-6	Decodifica dei caratteri del display	412
E-7	Cifre non decodificate del display	412
F-1	Simboli utilizzati	414
F-2	Listing della ROM	415
G-1	Segnali sui connettori	424
G-2	Collegamenti delle istruzioni di Restart	427

INTRODUZIONE ALL'EDIZIONE ITALIANA

Questo testo, proposto dalla Jackson Italiana nella sua linea di testi ad alto livello sulle problematiche sia teoriche che applicative dei microprocessori, si distacca da tutti i precedenti, non tanto perché li rende obsoleti, ma piuttosto perché affronta l'argomento in un modo del tutto diverso inquadrando didatticamente un momento operativo fondamentale, e cioè il collaudo e la messa a punto di sistemi a microprocessore.

Non a caso questo volume è stato realizzato dalla Hewlett-Packard, che, per la prima volta, getta tutto il peso di una grande industria operante nel settore della strumentazione elettronica e del calcolo sull'argomento microprocessori.

Due sono gli effetti di questa operazione culturale: la realizzazione di un'apparecchiatura didattica, il "Micro Lab", che arriva a comprendere in sé le problematiche della messa a punto dei sistemi a microprocessore, e la disponibilità di un "testo HP", intendendo con questo sottolineare la tradizionale eccellenza delle pubblicazioni della casa americana.

Questo "testo HP" presenta, a nostro giudizio, due chiavi di lettura tali da renderlo interessante anche a quanti provengono da precedenti letture sull'argomento microprocessori.

La prima chiave di lettura possiamo definirla tradizionale, in quanto il tecnico, sia inesperto che già operante nel settore, trova tutte le risposte possibili alla classica domanda "come usare un microprocessore?". Le diverse sezioni, Generalità, Hardware, Software, Ricerche guasti, soprattutto con riferimento alla novità dell'ultima, possono quasi rappresentare un definitivo inquadramento del bagaglio culturale del lettore sui microprocessori.

Ma la novità è la seconda chiave di lettura, in base alla quale il lettore vive le singole pagine del testo, verso un obiettivo, che possiamo pure chiamare la "ricerca guasti nel contesto microprocessore". Le modalità di esposizione sono tali, quindi, da permettere la definizione di un momento tecnico-culturale che sfocia verso uno dei maggiori e più costosi problemi della progettazione con sistemi a microprocessore, che fa capo al collaudo di quanto realizzato.

Nell'edizione italiana abbiamo cercato di mantenere il più possibile lo spirito del testo, per nulla sottrarre al lettore. Se ci saremo riusciti è anche merito della costante e precisa supervisione data a questo, come a molti altri libri della Jackson Italiana, da Aldo Cavalcoli e Cristina Cavenaghi e della preziosa collaborazione del Communication Department della Hewlett-Packard Italiana.

JACKSON ITALIANA EDITRICE

PREFAZIONE

INTRODUZIONE AL CORSO MICROPROCESSOR LAB

State per dedicare circa 50 ore del vostro tempo allo studio del corso Microprocessor Lab HP 5036A.
Come controparte potete aspettarvi di raggiungere i seguenti obiettivi:

a. Acquisire una conoscenza pratica dell'hardware di un sistema a microprocessore.
b. Acquisire una conoscenza base del software che viene utilizzato per controllare un sistema a microprocessore.
c. Imparare come il sistema utilizza tale software al fine di effettuare numerose e varie operazioni.
d. Utilizzare queste informazioni per apprendere delle tecniche pratiche di ricerca guasti applicabili ad ogni sistema a microprocessore.

Uno dei risultati più importanti che otterrete dallo studio di questo corso, sarà una confidenza legata all'aver compreso un sistema a microprocessore completo.
Tale confidenza chiarirà il mistero che è possibile utilizzare dei programmi per controllare l'hardware del sistema.
Potete così capire come un piccolo pezzo di silicio, detto microprocessore, è in grado di controllare un sistema. Quando avrete imparato come il software e il chip microprocessore esercitano tale controllo, per apprendere come opera qualunque altro sistema a microprocessore dovrete semplicemente impadronirvi dei dettagli specifici di quel certo sistema.
Le nuove tecniche di ricerca guasti che apprenderete saranno inoltre applicabili anche a sistemi che non siano basati su microprocessore. In effetti tali tecniche possono essere utilizzate per effettuare operazioni di ricerca guasti su qualunque strumento digitale.
Quando avrete completato tale corso perciò, applicando la conoscenza acquisita ad ogni problema che dovreste incontrare, dovreste essere in grado di capire e riparare qualunque sistema digitale.

CHE COSA CONTIENE IL CORSO MICROPROCESSOR LAB

Prima di iniziare, un breve cenno al contenuto e all'organizzazione del corso vi aiuterà a capire la materia che studierete. Una delle caratteristiche più importanti della struttura di questo corso è la modularità delle lezioni e delle sezioni. Poiché ogni studente parte con una diversa preparazione e con diversi obiettivi, tale struttura permette di seguire differenti metodologie di studio.

Se state studiando tale corso in una classe, sarà il vostro insegnante a scegliere tra le parti del corso quelle più adatte agli obiettivi della classe. Se invece studiate questo corso da autodidatti, leggete le descrizioni date qui di seguito in modo da vedere il contenuto delle varie lezioni. Leggete poi i diversi metodi di studio che vi sono proposti al termine di questa introduzione.

Confrontando la vostra preparazione attuale e i vostri obiettivi con tali metodi, potete fare in modo di ottenere il miglior risultato dal campo specifico che intendete studiare.

Il corso è suddiviso in sei sezioni, ciascuna delle quali, a seconda dei casi, è formata da una fino a cinque lezioni. Ogni lezione contiene esperimenti e domande che servono a rafforzare i concetti presentati e a mostrare delle applicazioni pratiche di quanto avete di volta in volta imparato.

SEZIONE I, NOZIONI FONDAMENTALI SUI MICROPROCESSORI.
È formata da tre lezioni che presentano una introduzione di base ai sistemi a microprocessore.

Lezione 1, Introduzione ai sistemi a microprocessore: dà delle informazioni introduttive sull'utilizzo delle applicazioni e sulla storia dei sistemi di calcolo in generale e dei sistemi a microprocessore in particolare.

Lezione 2, Sistemi di numerazione: presenta una introduzione ai sistemi di numerazione. Vengono trattati i sistemi di numerazione binaria, ottale, decimale ed esadecimale.

Lezione 3, Elementi di software: vi introduce ad alcuni concetti fondamentali della programmazione. Contiene anche una breve spiegazione della interazione tra hardware e software.

SEZIONE II, INTRODUZIONE ALLA PROGRAMMAZIONE.
È formata da tre lezioni che presentano una introduzione alla programmazione e alle istruzioni per utilizzare il Microprocessor Lab (μLab).

Lezione 4, Uso del Microprocessor Lab: contiene le indicazioni per introdurre dei dati, per far girare dei programmi sul μLab e per utilizzare le porte di ingresso/uscita.

Lezione 5, Alcuni concetti di software: discute altre tecniche di programmazione e alcune delle più sofisticate caratteristiche del μLab.

Lezione 6, All'interno del microprocessore: vi fa dare un breve sguardo all'interno del microprocessore, in modo da mostrare come vengono eseguiti i programmi. Questo vi aiuterà a capire come opera il sistema.

SEZIONE III, HARDWARE DEL SISTEMA A MICROPROCESSORE.
È formata da quattro lezioni che descrivono in dettaglio l'hardware dei sistemi a microprocessore.

Lezione 7, Concetti fondamentali di hardware: descrive l'hardware di un sistema base a microprocessore. Si insiste perché comprendiate le parti fondamentali di un sistema tipico.

Lezione 8, Decodifica degli indirizzi: descrive le proprietà e le caratteristiche dei circuiti di decodifica indirizzi del μLab. Vengono anche discussi altri tipi di circuiti di decodifica degli indirizzi.

Lezione 9, Memorie e periferiche: tratta le memorie e le periferiche utilizzate nel μLab. Discute inoltre altri tipi di hardware che sono utilizzati nei più svariati sistemi.

Lezione 10, Circuiti di controllo: tratta i segnali di controllo utilizzati nei sistemi a microprocessore. Vengono descritti i circuiti che generano, trasmettono e rispondono a tali segnali. Sono anche presentate delle considerazioni elettriche sui circuiti.

SEZIONE IV, SOFTWARE DEI MICROPROCESSORI.
È formata da cinque lezioni che trattano sofisticate tecniche e concetti di programmazione dei microprocessori.

Lezione 11, Registri e breakpoint: presenta degli argomenti introduttivi alla discussione più dettagliata che verrà fatta nel seguito della sezione. Sono descritti i registri del microprocessore e viene trattato l'uso dei breakpoint come strumento di collaudo software.

Lezione 12, Istruzioni dell'8085: descrive alcune nuove istruzioni per programmare l'8085. Ne viene presentato un gruppo rappresentativo, che comprende istruzioni logiche e aritmetiche.

Lezione 13, Tecniche di progettazione software: introduce alcune tecniche per progettare un sistema software complesso. Viene data enfasi al progettare del software che sia non solo efficiente, ma anche facile da collaudare e modificare.

Lezione 14, Controllo software delle periferiche: presenta il software utilizzato per controllare la tastiera e il display del μLab. Sono descritti i programmi che leggono dalla tastiera e scrivono sul display. Poiché la maggiore parte dei sistemi contengono una tastiera e un display, tali concetti possono essere applicabili a svariati tipi di sistemi a microprocessore.

Lezione 15, Rappresentazioni numeriche e algoritmi: discute alcune tecniche che sono utilizzate per calcolare complesse funzioni matematiche per un esteso intervallo numerico.

SEZIONE V, RICERCA DEI GUASTI NEI SISTEMI A MICROPROCESSORE.
È formata da quattro lezioni che introducono i concetti teorici relativi alla ricerca dei guasti e i nuovi strumenti e tecniche che sono stati sviluppati per riparare i sistemi a microprocessore.

Lezione 16, Strumenti portatili per la ricerca dei guasti: presenta l'utilizzo delle sonde logiche, dei generatori di impulsi logici e dei rivelatori di corrente.
Questi tre strumenti sono facili da usare ed assai efficaci in un vasto campo di situazioni di ricerca guasti. Vengono dati degli esempi pratici che mostrano come è possibile utilizzarli, singolarmente e in modo congiunto, al fine di localizzare dei guasti in sistemi a microprocessore.

Lezione 17, Analizzatori di firma e analizzatori logici: descrive e mostra come vengono usati questi due strumenti specializzati al fine di cercare guasti in sistemi a microprocessore. Vengono illustrate delle applicazioni pratiche di assistenza sul campo, in produzione, e nello sviluppo dei prodotti.

Lezione 18, Ricerca di guasti nei sistemi a microprocessore: spiega la filosofia e le metodologie di ricerca guasti nel caso di sistemi a microprocessore. Sono discussi dei problemi particolari e le loro relative soluzioni.

Lezione 19, Ricerca di guasti nel Microprocessor Lab: descrive l'utilizzo del diagramma di flusso per la ricerca guasti del μLab. Per prima cosa, utilizzando

tale diagramma di flusso, viene trovato un guasto; vi viene poi mostrato come inserire nel sistema diversi guasti e come localizzarli.

SEZIONE VI, UNA PANORAMICA DEGLI ALTRI MICROPROCESSORI.
È formata da una sola lezione. Presenta una panoramica di numerosi microprocessori attualmente disponibili.

Lezione 20, Panoramica dei microprocessori: descrive altri tipi di microprocessori ed evidenzia che, benché i dettagli varino da un microprocessore ad un altro, i concetti essenziali rimangono invariati.

Oltre alle nozioni contenute nelle lezioni, il corso comprende anche le seguenti otto appendici, che si trovano nella Sezione VII e che vi danno numerose informazioni di riferimento:

A. Soluzione dei problemi. Contiene le risposte di tutte le domande e le soluzioni di tutti i problemi di programmazione e di ricerca guasti.
B. Le istruzioni dell'8085 A. Contiene una descrizione dell'insieme completo di istruzioni dell'8085 A, dandone i relativi codici esadecimali.
C. Tabelle delle firme. Contiene le informazioni delle firme di tutti i punti di test e di tutti i piedini dei circuiti integrati del μLab.
D. Lettura degli schemi elettrici. Contiene una breve descrizione di livello logico utilizzati nei simboli logici.
E. Programmi dimostrativi e di utilità. Contiene la descrizione di tali programmi e le indicazioni per il loro utilizzo.
F. Listing della ROM del Microprocessor Lab. Contiene il listing completo del contenuto della ROM del μLab.
G. Espansione del Microprocessor Lab. Contiene una breve descrizione dei segnali di ingresso e di uscita dei connettori del μLab e dà delle informazioni su come utilizzare teli segnali.

Tra il materiale di riferimento sono anche inclusi un esteso glossario e una bibliografia, entrambi orientati alle caratteristiche dei sistemi a microprocessore.

L'ultima pagina del manuale contiene lo schema elettrico del μLab. L'interno della copertina contiene poi i messaggi di errore del sistema. La copertina ancora, aperta, vi presenta dei riferimenti veloci per il μLab e vi dà delle brevi definizioni dei tasti e degli interruttori.

PREPARAZIONE E PROCEDURE DI STUDIO PROPOSTE

La preparazione minima che è richiesta per il corso sul μLab prevede una comprensione dei circuiti digitali fondamentali e della loro simbologia. Se sentite il bisogno di ripassare tali argomenti il corso "HP's Practical Digital Electronics Course", potrà aiutarvi.

Se desiderate una conoscenza generale del software, dell'hardware e delle tecniche di ricerca guasti, la cosa migliore che potete fare è quella di partire dall'inizio del corso e procedere quindi sequenzialmente una lezione dopo l'altra.

Se volete studiare alcune, ben precise, parti del corso, potete naturalmente modificare tale sequenza.

La figura seguente vi mostra quali lezioni devono essere studiate per ciascuno di questi tre argomenti.

a. Hardware
b. Software
c. Ricerca Guasti.

L'ombreggiatura pesante indica le lezioni assolutamente necessarie, mentre quella leggera indica le lezioni facoltative. Così, la Lezione 7, presenta una panoramica sull'hardware dei microprocessori. Se siete interessati ad apprendere il software, lo studio della Lezione 7 non sarà obbligatorio ma vi darà comunque alcune eccellenti nozioni di base.

	Hardware	Software	Ricerca guasti
I. Nozioni fondamentali sui microprocessori			
Lezione 1. Introduzione ai sistemi a microprocessore			
Lezione 2. Sistemi di numerazione			
Lezione 3. Elementi di software			
II. Introduzione alla programmazione			
Lezione 4. Uso del Microprocessor Lab			
Lezione 5. Alcuni concetti di software			
Lezione 6. All'interno del microprocessore			
III. Hardware del sistema a microprocessore			
Lezione 7. Concetti fondamentali di hardware			
Lezione 8. Decodifica degli indirizzi			
Lezione 9. Memorie e periferiche			
Lezione 10. Circuiti di controllo			
IV. Software dei microprocessori			
Lezione 11. Registri e breakpoint			
Lezione 12. Istruzioni dell'8085			
Lezione 13. Tecniche di progettazione software			
Lezione 14. Controllo software delle periferiche			
Lezione 15. Rappresentazioni numeriche e algoritmi			
V. Ricerca dei guasti nei sistemi a microprocessore.			
Lezione 16. Strumenti portatili per la ricerca dei guasti			
Lezione 17. Analizzatori di firma e analizzatori logici			
Lezione 18. Ricerca guasti nei sistemi a microprocessore			
Lezione 19. Ricerca guasti nel Microprocessor Lab			
VI. Una panoramica degli altri microprocessori			
Lezione 20. Panoramica dei microprocessori			

Legenda

Indispensabile — Facoltativo ma opportuno — Non richiesto

Guida per lo Studio del Microprocessor Lab.

I
NOZIONI FONDAMENTALI SUI MICROPROCESSORI

L'introduzione dei microprocessori ha provocato un cambiamento sostanziale nel progetto dei sistemi logici. Nell'approccio tradizionale, a logica sparsa o cablata, i sistemi vengono progettati in funzione dell'applicazione richiesta, utilizzando blocchi logici elementari (flip-flop, gate, contatori ecc.) i quali sono composti e collegati in modo da ottenere il flusso di dati richiesto. Nell'approccio della logica cablata, ogni applicazione richiede un progetto diverso, e fra i differenti sistemi, in genere, esiste solo una limitata similarità. Questo tipo di approccio ricorda molto quello che si ha nei circuiti analogici in cui la struttura circuitale è strettamente legata alla funzione operativa richiesta. Risulta così difficile modificare la funzione del circuito, una volta che questo sia stato realizzato.

Viceversa i microprocessori costituiscono dei sistemi di controllo di utilizzo generale che, con limitate modifiche circuitali, possono essere adattati ad una vasta gamma di applicazioni. L'individualità dei differenti sistemi è presente solo nella sequenza di istruzioni (il *programma*) che controllano le operazioni del sistema. Possiamo quindi identificare due diversi aspetti presenti nei sistemi a microprocessori: i componenti fisici (detti, nel loro complesso, *hardware*) e i programmi (*software*).

In queste sezioni viene dapprima introdotto l'hardware che costituisce un sistema a microprocessore tipico.
Successivamente vengono discussi i concetti software e i linguaggi di programmazione. Infine viene presentato un esempio di come hardware e software interagiscono per realizzare un dispositivo logico.

LEZIONE 1

Introduzione ai sistemi a microprocessore

INTRODUZIONE

Il microprocessore è un grosso e complesso circuito integrato (Integrated Circuit, IC) che, funzionalmente, contiene tutta la circuiteria di calcolo e controllo di un minicalcolatore e attraverso cui è possibile inserire una capacità di calcolo, a «buon mercato», in molti dispositivi: voltmetri, forni a microonde, registratori di cassa e persino giochi. In questa lezione sono descritti il modo generale di operare dei sistemi a microprocessore e i blocchi funzionali che li costituiscono.

LO SVILUPPO DEL MICROPROCESSORE

I primi calcolatori elettronici furono costruiti utilizzando migliaia di valvole. Queste macchine erano enormi e inaffidabili, e erano per lo più una curiosità di laboratorio. La generazione che seguì fu costruita con transistor, grazie ai quali i computer si fecero più affidabili, più piccoli ed economici. Queste macchine «a stato solido», rappresentarono la nascita del calcolatore come dispositivo praticamente utilizzabile.

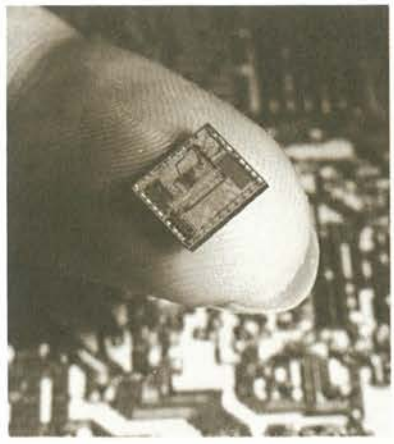
Macchina da calcolo ENIAC. *Costruita nel 1946, fu la prima macchina da calcolo elettronica non specializzata. Conteneva oltre 18.000 valvole ed aveva bisogno di un alimentatore grande quanto la metà del calcolatore stesso. (UPI)*

All'inizio degli anni 60 furono costruiti calcolatori più potenti che facevano uso di centinaia di gate, flip-flop, ed altri circuiti integrati a *bassa scala d'integrazione* (Small Scale Integration, SSI). Grazie allo sviluppo della tecnologia dei semiconduttori, diventò possibile inserire in un unico IC dozzine di gate. Contatori, decoders (decodificatori), registri e sommatori sono esempi di IC a *media scala d'integrazione* (Medium Scale Integration, MSI).

Calcolatrice da tavolo programmabile. *Entrata sul mercato nel 1968, è più potente del ben più grosso ENIAC: È costruita con transistor.*

Questa tendenza alla miniaturizzazione continuò, e nel 1971 fu prodotto il primo microprocessore (il 4004). I microprocessori, su di un unico chip di circuito integrato, contengono le principali sezioni di controllo e di calcolo proprie di un calcolatore, il cui insieme è detto *unità centrale di processo* (Central Processing Unit, CPU). I microprocessori sono spesso chiamati *micro-unità di processo* (Micro Processing Unit, MPU). Un chip di microprocessore contiene migliaia di gate ed è perciò detto un dispositivo a *larga scala d'integrazione* (Large Scale Integration, LSI). Contemporaneamente furono sviluppati anche dispositivi LSI di memoria, che immagazzinano migliaia di bit di informazione digitale su di un solo IC. Grazie a questi due tipi di dispositivi LSI fu possibile ridurre enormemente le dimensioni ed il costo di piccoli calcolatori. I microprocessori inoltre hanno reso conveniente l'inserire dei calcolatori specializzati in molti prodotti piccoli e a basso costo.

Calcolatrice tascabile programmabile. *La tecnologia dei circuiti integrati ha reso possibile costruire una calcolatrice tascabile più potente della calcolatrice da tavolo della precedente fotografia.*



Microcalcolatore su di un singolo chip, *di circa mezzo centimetro di lato. Questo piccolo chip contiene virtualmente tutti i circuiti elettronici di un piccolo sistema calcolatore. (Intel Corp.)*

PERCHÉ VENGONO USATI I MICROPROCESSORI

I microprocessori vengono attualmente impiegati in molti prodotti precedentemente costruiti in logica cablata. I progetti basati su microprocessori sono di solito meno costosi e sono costituiti da un minor numero di componenti rispetto ai progetti che sostituiscono. Piccoli sistemi basati su microprocessori possono essere realizzati utilizzando solo uno o due IC (per un costo complessivo inferiore a dieci dollari), spesso in sostituzione di schede formate da dozzine di IC meno complessi. La significativa riduzione del numero dei componenti discreti e delle interconnessioni migliora inoltre l'affidabilità del sistema.

Questa riduzione delle dimensioni può anche essere ottenuta utilizzando, al posto dei microprocessori, circuiti integrati «custom» (cioè specializzati per una applicazione). Il progetto di un IC custom tuttavia può essere estremamente complesso e costoso e spesso infatti si hanno dei costi ben superiori ai 100.000 dollari. Questa spesa può essere evidentemente giustificata solo in caso di progetti che debbano essere prodotti in grande quantità, dove cioè il costo dello sviluppo possa essere suddiviso tra molte migliaia di prodotti. Con i microprocessori, cioè con IC standard, si possono raggiungere gli stessi livelli di miniaturizzazione. La specializzazione è ottenuta attraverso un programma «custom» inserito in memorie standard, la cui produzione è un processo relativamente economico.

La flessibilità e la potenza dei sistemi basati su microprocessori rende possibile inserire, negli stessi sistemi, molte caratteristiche sofisticate, che nel passato erano difficilmente ottenibili. Ad esempio, sistemi con microprocessori possono spesso autocollaudarsi per buona parte, e in caso di cattivo funzionamento fornire opportuni messaggi di errore. In strumenti tipo voltimetri digitali possono essere inserite delle funzioni aritmetiche (per esempio, il calcolo automatico della media su un certo numero di misure), la possibilità di sommare o sottrarre ad ogni misura un valore di offset, e funzioni di autotaratura. I microprocessori rendono anche più semplice utilizzare, per i pannelli di comando, le tastiere al posto dei commutatori. Un'altra caratteristica interessante è il fatto di poter essere facilmente e completamente controllati da dispositivi lontani (Remote Control).

Voltmetro digitale basato su microprocessore. *Il microprocessore effettua alcune funzioni tra cui la taratura automatica, un'operazione di media tra un certo numero di misure e la somma automatica di un valore di offset.*

Tra i prodotti cresciuti grazie alla tecnologia dei microprocessori, uno dei più diffusi è il registratore di cassa elettronico o *terminale per punti di vendita* (Point of Sale, POS). Oltre a sostituire le vecchie macchine, di cui è stata migliorata l'affidabilità, sono diventate possibili nuove prestazioni. L'inventario può essere aggiornato automaticamente, assegnando ad ogni articolo da vendere un numero di codice che l'operatore introduce battendo sulla tastiera o leggendolo dall'etichetta presente nel prodotto attraverso un lettore ottico. Possono essere aggiunte automaticamente tasse di vendita. È possibile persino inserire il numero di conto corrente dei clienti, in modo che il denaro sia automaticamente trasferito dal conto del cliente a quello del magazzino.

Le bilance elettroniche ci danno un altro esempio delle caratteristiche che si possono trovare nei prodotti basati sui microprocessori. L'operatore inserisce il prezzo, al chilogrammo, di un prodotto, e la bilancia lo pesa e ne visualizza il peso ed il prezzo. La bilancia può anche sottrarre automaticamente il peso del contenitore (la tara).

Bilancia basata su microprocessore. *L'utente introduce il prezzo unitario e la bilancia calcola il costo totale. La tara può essere sottratta in modo automatico. (Toledo Scala)*

Prodotti con tutte queste caratteristiche potrebbero essere realizzati anche senza l'impegno dei microprocessori, ma sarebbero così complessi e costosi che risulterebbero ben poco pratici. I sistemi con microprocessori sono così potenti perché ogni IC è estremamente complesso. Il progettista di un prodotto non deve perciò preoccuparsi dei dettagli costruttivi del microprocessore e, contemporaneamente, sono evitati i problemi legati alle dimensioni e all'affidabilità, propri dei sistemi complessi. Inoltre essendo l'operatività di un prodotto soprattutto affidata al software, diventa semplice aggiornare o modificare il progetto. Il microprocessore ha veramente rivoluzionato il progetto di molti prodotti.

UN SISTEMA BASE CON MICROPROCESSORE

Consideriamo un sistema con tastiera e display (visualizzatore) numerico, per esempio un calcolatore tascabile. Quando viene premuto un tasto, sul display dovrebbe apparire il numero corrispondente al tasto premuto. Questo sistema è una applicazione molto ovvia di un microprocessore, ed è per molti aspetti simile al Microprocessor/Lab.

La Figura 1-1 mostra lo schema a blocchi di un sistema che realizza queste funzioni. Il microprocessore (detto anche processore) è il cervello del sistema. Contiene tutta la logica necessaria per riconoscere ed eseguire la sequenza di istruzioni che costituiscono il programma, immagazzinato, eventualmente insieme ai dati, in memoria. La fotografia al termine del libro (nel risvolto di copertina) mostra questi componenti sulla scheda del Microprocessor Lab (μLab).

Il microprocessore ha bisogno di scambiare informazioni con la tastiera e il display. La *porta d'ingresso* (Input Port), dalla quale il microprocessore può leggere dati, collega il processore con la tastiera. La *porta d'uscita* (Output Port), alla quale il microprocessore può inviare dati, collega invece il processore con il display.

All'interno del microcalcolatore i blocchi sono collegati da tre bus. Un *bus* è un gruppo di linee (fili) che collegano, in modo parallelo, i dispositivi del sistema. Il microprocessore si serve del *bus degli indirizzi* (Address Bus) per selezionare le locazioni di memoria o le porte d'ingresso e uscita. Gli indirizzi sono un po' come i numeri delle caselle postali: identificano in quali locazioni devono essere messe delle informazioni o da quali locazioni le informazioni devono essere prelevate.

Il microprocessore, una volta che ha selezionato, attraverso il bus degli indirizzi, una certa locazione, trasferisce il dato attraverso il *bus dei dati* (Data Bus). La direzione del trasferimento dell'informazione può essere dal processore alla memoria o ad una porta d'uscita, oppure viceversa dalla memoria o da una porta d'ingresso verso il processore. Notate che il microprocessore è comunque coinvolto in tutti i trasferimenti dei dati; i dati di solito non passano direttamente da una porta all'altra o dalla memoria ad una porta.

Il terzo bus è detto *bus di controllo* (Control Bus) ed è costituito da un gruppo di segnali di cui il microprocessore si serve per segnalare, alla memoria e ai dispositivi di I/O (Input - Output), di essere pronto ad effettuare il trasferimento di un dato. Alcuni segnali del bus di controllo permettono invece ai dispositivi di memoria o di I/O di poter inviare certe richieste al processore. Sulla scheda del μLab il bus di controllo non è visibile in quanto è direttamente collegato alla logica di controllo, la quale, a sua volta, genera i segnali di controllo per i diversi dispositivi del sistema.

Il *bit* (abbreviazione di Binary Digit, cifra binaria) è una cifra d'informazione binaria (1 o 0). Un segnale digitale (basso o alto) ha associato un bit d'informazione. I dati, che i microprocessori trattano, non sono i bit presi singolarmente, ma sono formati da gruppi di bit detti *parole* (Words). I microprocessori attualmente più diffusi fanno uso di parole di otto bit: una parola di questa dimensione è detta *byte*. Tali microprocessori sono perciò processori ad otto bit; in tale caso quindi le dizioni *byte* e *parole* sono del tutto equivalenti. Fate comunque attenzione che la dizione *parole* può essere anche usata per indicare gruppi di sedici o più bit.

I bus degli indirizzi e dei dati possono essere ritrovati anche nella scheda del μLab. Il bus dei dati è costituito da otto linee, mentre il bus degli indirizzi da sedici.

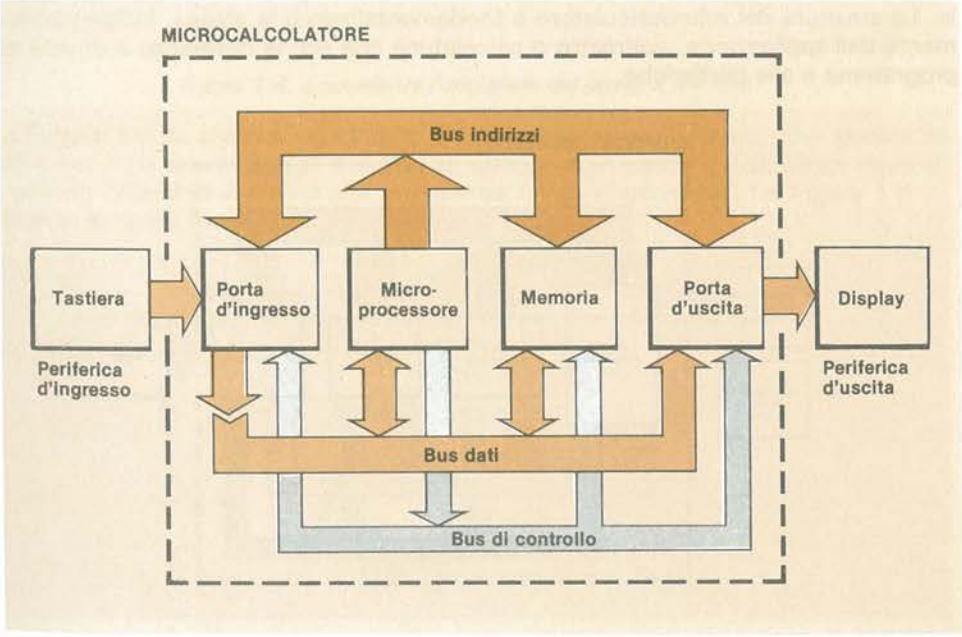


Figura 1-1. Sistema base a microprocessore

I PROGRAMMI

Per far sì che il sistema esegua una certa funzione, è richiesta una ben precisa sequenza di istruzioni. Ad esempio:

1. Leggi il dato dalla tastiera.
2. Visualizza il dato.
3. Ripeti l'operazione (ritorna al passo 1).

Affinché, data una certa sequenza di istruzioni, il microprocessore possa svolgere il compito voluto, è necessario che le istruzioni vengano tradotte in un codice comprensibile al microprocessore stesso. Tali codici vengono poi immagazzinati nella memoria del sistema. Il microprocessore inizierà leggendo dalla memoria la prima istruzione codificata. Il microprocessore decodifica il contenuto dell'istruzione ed effettua l'operazione indicata; dalla successiva locazione di memoria, il microprocessore legge quindi un'altra istruzione ed effettua l'operazione corrispondente. Il processo si ripete, una locazione di memoria dopo l'altra.

Alcune istruzioni fanno sì che il microprocessore, eseguendo un *salto*, esca da tale sequenza e prelevi l'istruzione successiva da una locazione di memoria diversa. Il microprocessore può, ad esempio, ricevere dal programma l'indicazione di ritornare ad una istruzione precedente del programma stesso, generando così un *loop* (cioè un anello) che può essere eseguito più volte. In questo modo, utilizzando un programma relativamente breve, è possibile eseguire delle operazioni che devono essere ripetute numerose volte.

PERIFERICHE

Un sistema completo a microprocessore, costituito dal microprocessore, dalla memoria, e dalle porte d'uscita e d'ingresso, è detto *microcalcolatore* (Microcomputer). I dispositivi collegati alle porte d'ingresso e di uscita (ad esempio la tastiera e il display) sono detti *periferiche* (Peripherals), ovvero dispositivi *ingresso/uscita* (di I/O, Input/Output, Ingresso/Uscita). Le periferiche sono l'interfaccia tra il sistema e l'utilizzatore, ma possono anche servire per collegare il microcalcolatore ad altri sistemi. Si parla di periferiche anche quando ci si riferisce a dispositivi di memoria, come le unità a disco o a nastro.

Nel campo della strumentazione, un esempio di applicazione dei microprocessori è dato dal voltmetro digitale a microprocessore (Figura 1-2). Le sue periferiche d'ingresso sono un convertitore analogico-digitale e gli interruttori che selezionano la scala e la funzione. La periferica d'uscita è un display (visualizzatore) digitale. La struttura del microcalcolatore è fondamentalmente la stessa, indipendentemente dall'applicazione, voltmetro o calcolatrice che sia; la differenza è dovuta al programma e alle periferiche.

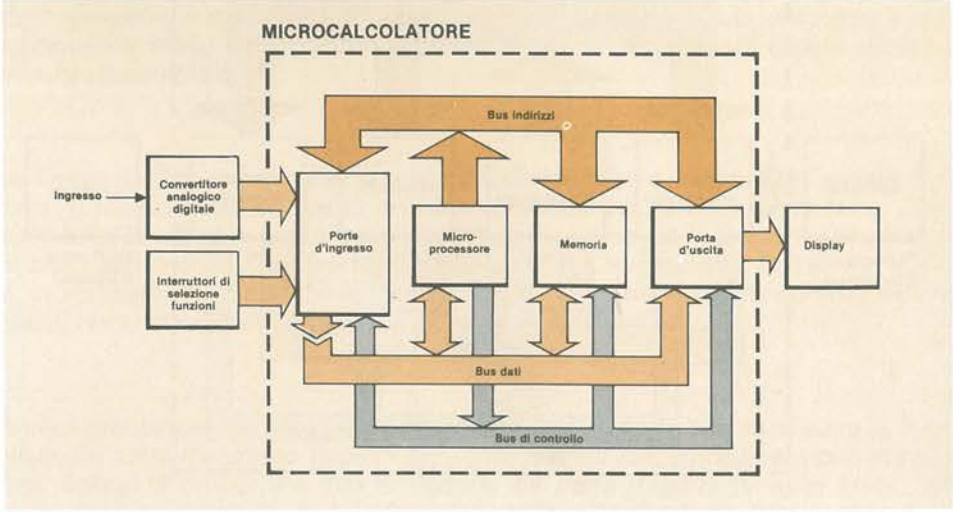
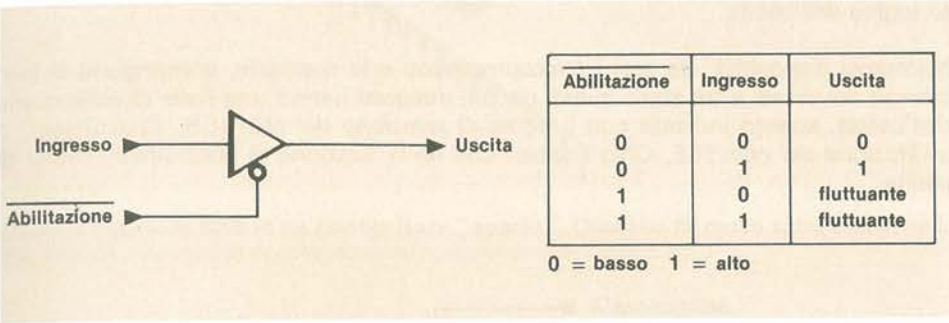


Figura 1-2. Voltmetro digitale basato su microprocessore

DISPOSITIVI A TRE STATI

Nel sistema a microprocessore tutti i dispositivi scambiano informazioni con il microprocessore servendosi delle *stesse* linee (il bus dei dati). Il microprocessore seleziona ad accedere al bus dei dati il dispositivo con cui intende comunicare e scollega tutti gli altri. È la capacità, che hanno i dispositivi collegati al bus, di avere un'uscita a *tre stati* (Three-State) che permette al microprocessore di condizionare un dispositivo ad essere attivo (acceso o inattivo, spento, scollegato).



$\overline{\text{Abilitazione}}$	Ingresso	Uscita
0	0	0
0	1	1
1	0	fluttuante
1	1	fluttuante

0 = basso 1 = alto

Figura 1-3. Il driver a tre stati

La Figura 1-3 riporta la rappresentazione simbolica e la tabella della verità di un dispositivo a tre stati (three-state buffer, anche detto *three-state driver*). Tale dispositivo, oltre al solito ingresso e alla solita uscita, dispone anche di un'abilitazione (Enable) dell'uscita. Quando il segnale di abilitazione è basso, il dispositivo si comporta come un buffer qualsiasi: il segnale d'ingresso è trasferito pari pari in uscita. Quando il segnale di abilitazione è invece alto, l'uscita del dispositivo è di fatto scollegata, aperta.

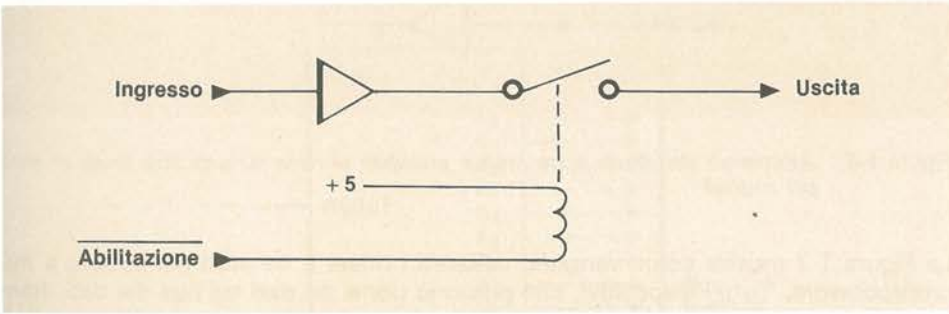


Figura 1-4. Equivalente funzionale del driver a tre stati

La Figura 1-4 vi mostra un circuito funzionalmente equivalente, che genera lo stato «aperto» servendosi di un relé. Lo stato in cui l'uscita è disabilitata (aperta) è spesso chiamato *stato ad alta impedenza* (High Impedance). La Figura 1-5 riporta lo schema elettrico di un tipico buffer a tre stati.

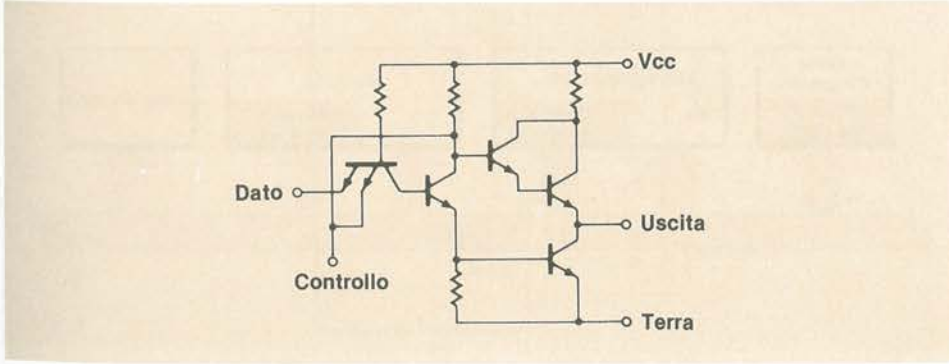


Figura 1-5. Schema di una tipica uscita a tre stati

L'importanza dei driver a tre stati è legata al fatto che, grazie ad essi, diventa possibile che più dispositivi «spartiscano» una stessa linea di dato. Così, nel circuito della Figura 1-6, ciascuno dei tre dispositivi è in grado di pilotare l'unica uscita. È sufficiente che, delle linee di abilitazione, solo una sia di volta in volta bassa: il dispositivo corrispondente piloterà l'uscita. Se più di uno fosse abilitato, più dispositivi cercherebbero contemporaneamente di pilotare l'uscita. Questa condizione non è permessa in quanto non è possibile in tal caso prevedere lo stato logico dell'uscita.

Numerosi dispositivi, tra essi i microprocessori e le memorie, contengono al loro interno dei driver a tre stati: questi circuiti integrati hanno una linea di abilitazione dell'uscita, spesso indicata con il nome di *selezione del chip* (CS, Chip Select) o *abilitazione del chip* (CE, Chip Enable) che ha la funzione di controllare i driver di uscita.

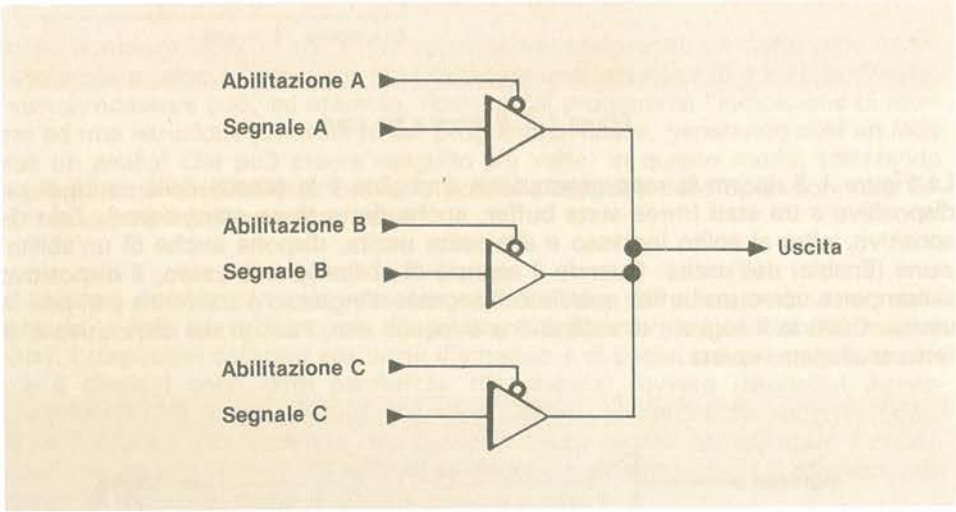


Figura 1-6. Attraverso dei driver a tre stati è possibile portare su una sola linea di dato più segnali

La Figura 1-7 mostra come vengono utilizzati i driver a tre stati nei sistemi a microprocessore. Tutti i dispositivi, che possono porre dei dati sul bus dei dati, hanno sulle loro uscite dei driver a tre stati. Il microprocessore genera dei segnali di controllo (appartenenti al bus di controllo), in modo da abilitare i driver a tre stati del dispositivo da cui intende leggere un dato. Sono invece disabilitati i driver a tre stati di tutti gli altri dispositivi.

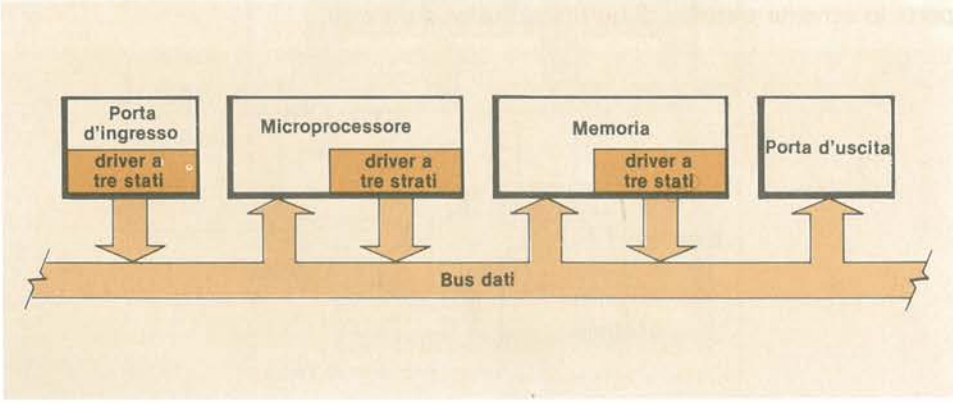
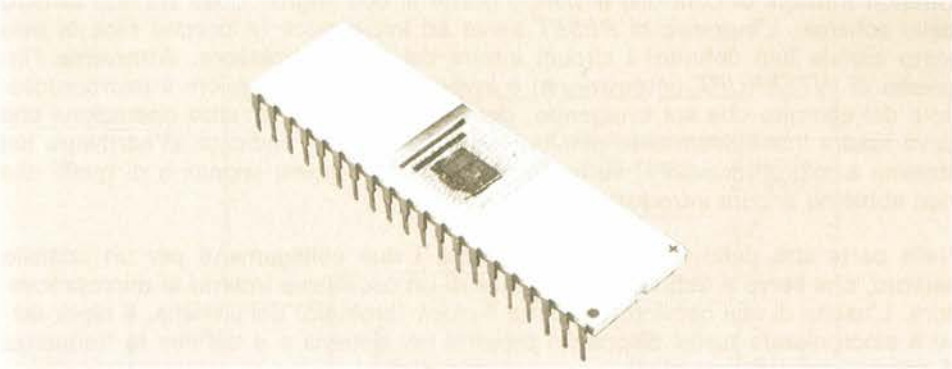


Figura 1-7. Driver a tre stati in un sistema a microprocessore



Microprocessore 8085 in un contenitore "aperto". *Quaranta fili molto sottili collegano le piazzole sul chip ai punti di collegamento sul contenitore. (Intel Corp.)*

IL MICROPROCESSORE

Nella Figura 1-8 sono presentati i principali segnali di collegamento di un microprocessore tipico. Sono presenti sedici uscite di indirizzo che formano il bus degli indirizzi e otto piedini di dato che si collegano al bus dei dati. I piedini di dato sono *bidirezionali,* intendendo con questo che attraverso tali piedini i dati possono entrare o uscire. Le linee *READ* (lettura) e *WRITE* (scrittura) sono i segnali di controllo che coordinano il movimento dei dati sul bus dei dati.

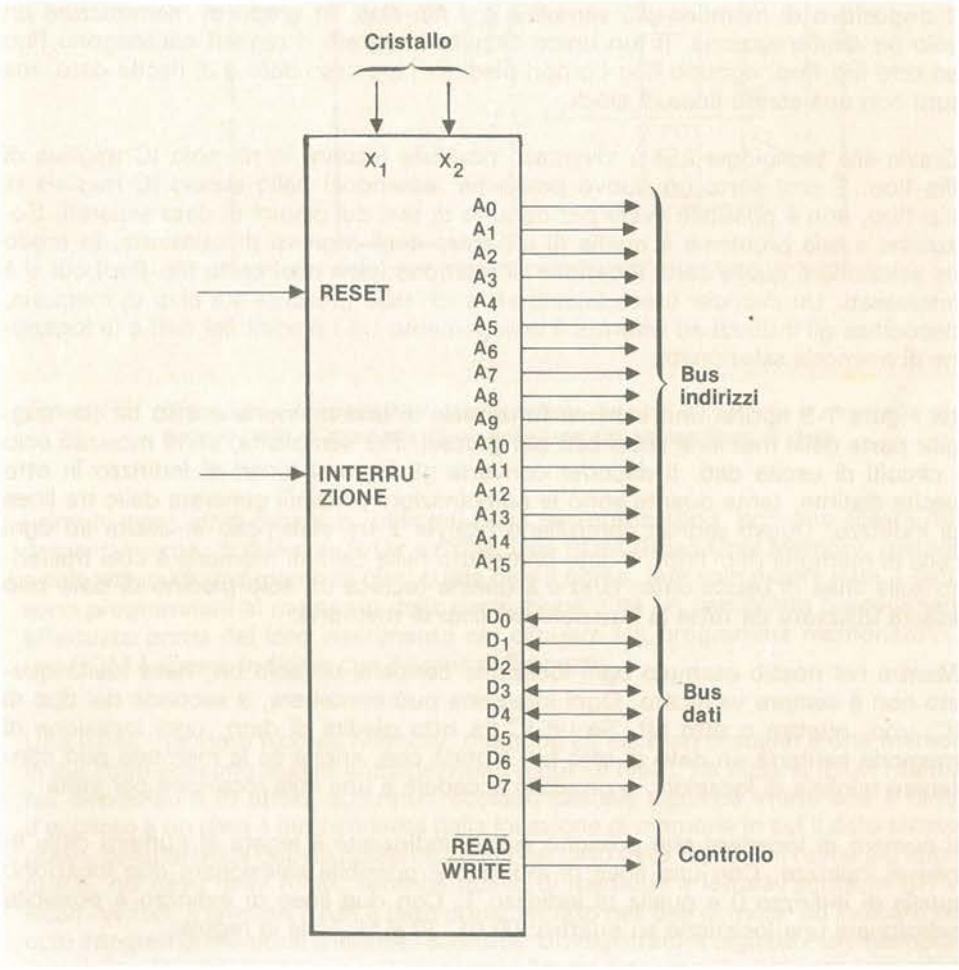


Figura 1-8. I segnali principali di un microprocessore

Ulteriori funzioni di controllo si hanno grazie ai due segnali posti sul lato sinistro dello schema. L'ingresso di *RESET* serve ad inizializzare (a portare cioè in uno stato iniziale ben definito) i circuiti interni del microprocessore. Attraverso l'ingresso di *INTERRUPT* (interruzione) è invece possibile distogliere il microprocessore dal compito che sta svolgendo, per fargli effettuare un'altra operazione che deve essere immediatamente gestita. Nella Sezione III, dedicata all'hardware del sistema a microprocessore, verrà discusso l'uso di questi segnali e di quelli che non abbiamo ancora introdotto.

Nella parte alta dello schema sono indicati i due collegamenti per un cristallo esterno, che serve a definire la frequenza di un oscillatore interno al microprocessore. L'uscita di tale oscillatore è detta il *clock* (orologio) del sistema. Il clock serve a sincronizzare tutti i dispositivi presenti nel sistema e a definire la frequenza di esecuzione delle istruzioni.

MEMORIE

Per immagazzinare programmi e dati, i sistemi a microprocessore fanno uso, di solito, di memorie a circuito integrato, in grado di immagazzinare, su un solo dispositivo, moltissimi bit di dati. Attualmente sono disponibili dei dispositivi in grado di contenere più di 65.000 bit sullo stesso chip. Una memoria di 65.536 bit può immagazzinare più di ottomila caratteri alfanumerici, cioè circa tre pagine di un libro come questo, su di una piastrina di silicio di area inferiore a mezzo centimetro quadrato.

Il dispositivo di memoria più semplice è il *flip-flop,* in grado di memorizzare un solo bit d'informazione. In un unico circuito integrato, i registri contengono fino ad otto flip-flop, ognuno con i propri piedini di ingresso dato e di uscita dato, ma tutti con una stessa linea di clock.

Grazie alla tecnologia LSI è diventato possibile inserire in un solo IC migliaia di flip-flop. È così sorto un nuovo problema: essendoci nello stesso IC migliaia di flip-flop, non è possibile avere per ognuno di essi dei piedini di dato separati. Soluzione a tale problema è quella di utilizzare degli ingressi di selezione, in modo da selezionare quella certa locazione di memoria (cioè quel certo flip-flop) cui si è interessati. Un *decoder* (decodificatore), anch'esso presente sul chip di memoria, decodifica gli indirizzi ed effettua il collegamento tra i piedini dei dati e la locazione di memoria selezionata.

La Figura 1-9 riporta uno schema funzionale di una memoria a otto bit (la maggior parte delle memorie sono ben più grosse). Per semplicità, sono mostrati solo i circuiti di uscita dati. Il decoder converte gli ingressi binari di indirizzo in otto uscite distinte, tante quante sono le combinazioni possibili generate dalle tre linee di indirizzo. Questi segnali controllano i driver a tre stati posti in uscita ad ogni cella di memoria (flip-flop). Il dato contenuto nella cella di memoria è così trasferito sulla linea di uscita dato. Grazie a questa tecnica un solo piedino di dato può essere utilizzato da tutte le locazioni del chip di memoria.

Mentre nel nostro esempio ogni locazione contiene un solo bit, nella realtà questo non è sempre verificato. Ogni locazione può contenere, a seconda del tipo di IC, uno, quattro o otto bit. Se un IC ha otto piedini di dato, ogni locazione di memoria conterrà un dato di otto bit. Notate che, anche se la memoria può contenere migliaia di locazioni, è possibile accedere a una sola locazione per volta.

Il numero di locazioni che possono essere indirizzate è legata al numero delle linee di indirizzo. Con una linea di indirizzo è possibile selezionare due locazioni: quella di indirizzo 0 e quella di indirizzo 1. Con due linee di indirizzo è possibile selezionare una locazione su quattro: 00,01, 10 e 11. Vale la regola:

Numero di locazioni $= 2^N$
In cui N = numero delle linee di indirizzo

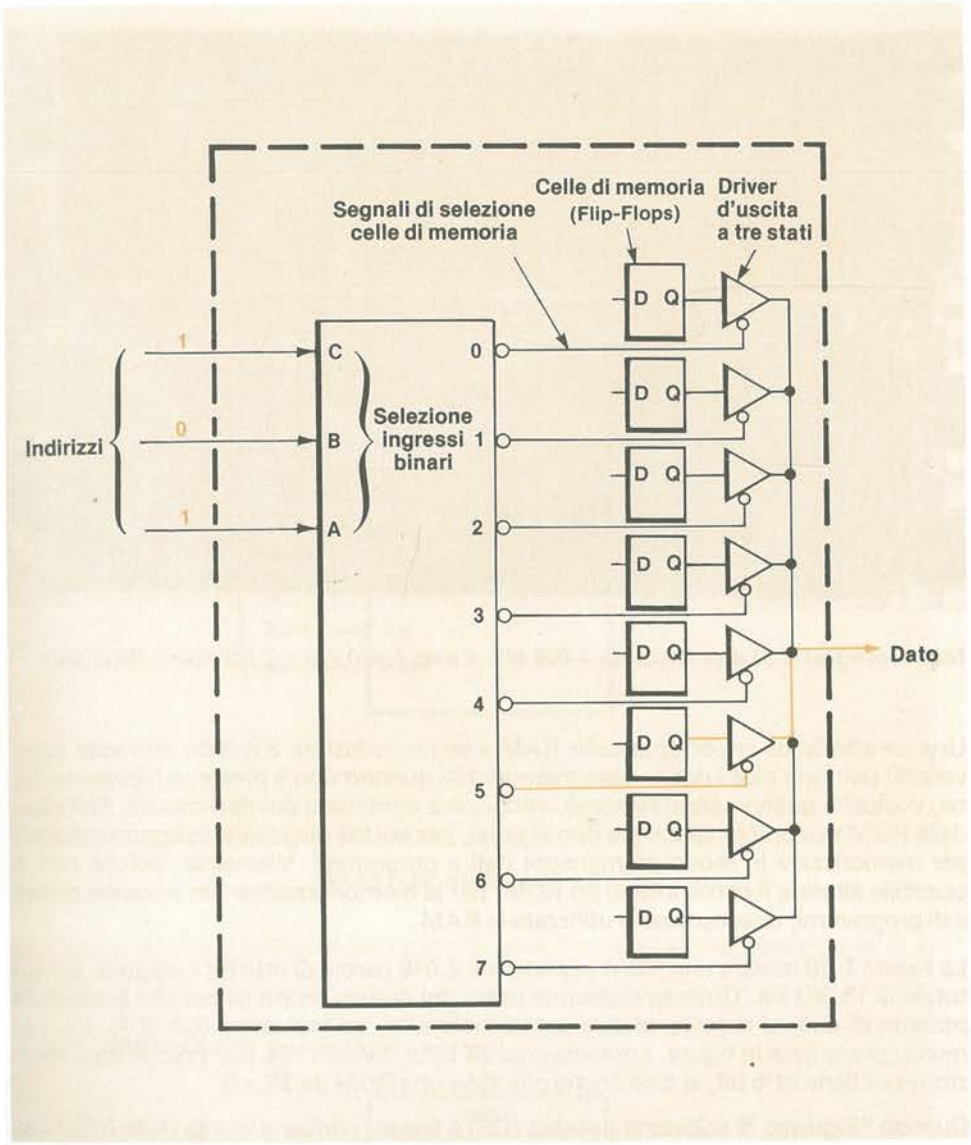
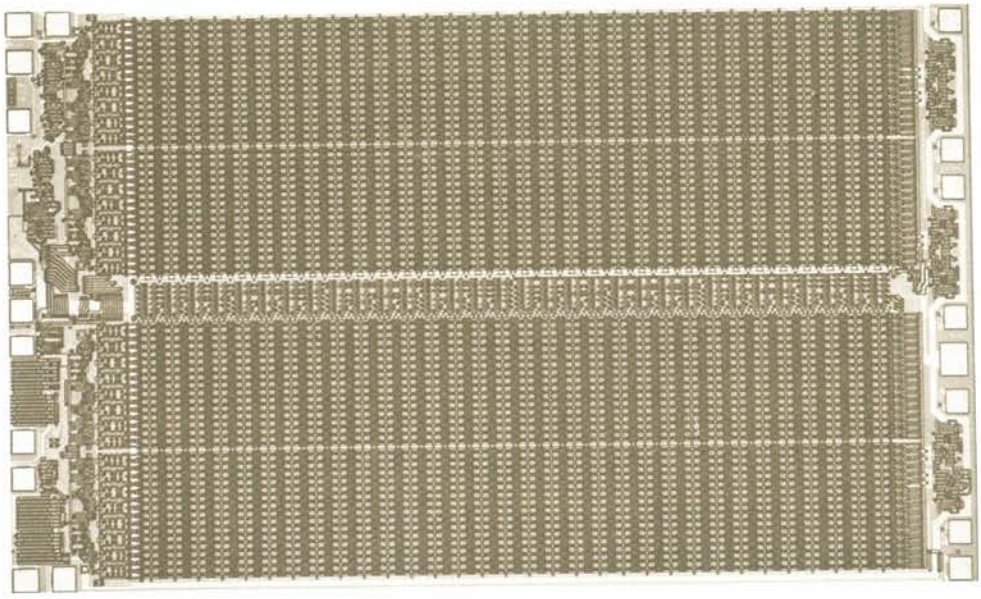


Figura 1-9. Diagramma funzionale di una memoria da otto bit, in cui è mostrato come tutte le celle di memoria siano collegate alla stessa linea di dato.

RAM E ROM

I circuiti integrati di memoria, utilizzati con i microprocessori, possono essere, fondamentalmente, di due tipi: *RAM* e *ROM*. Una ROM (Read Only Memory, memoria a sola lettura) è una memoria che, come dice il nome, può solo essere letta. I dati vi sono programmati al momento della costruzione o da un particolare procedimento effettuato prima del loro inserimento nel circuito. Un programma memorizzato in una ROM è spesso indicato con il nome di *firmware*.

Una RAM (Random Access Memory, memoria ad eccesso casuale) è una memoria in cui i dati possono essere immagazzinati e quindi riletti. La dizione RAM riferita a tali dispositivi è in effetti scorretta: accesso casuale significa infatti che il tempo d'accesso a un dato è indipendente dalla locazione di memoria in cui il dato stesso è posto, caratteristica questa presente anche nel caso delle ROM. Un nome più appropriato, nel caso delle RAM, sarebbe quello di memorie a lettura / scrittura (R / W, Read / Write): il termine RAM è però ormai entrato nell'uso comune ad indicare i circuiti integrati di memoria a lettura / scrittura. Un registratore digitale è un esempio di memoria ad accesso non casuale, in quanto il tempo necessario per accedere ad una certa locazione è in funzione della posizione del nastro.



Microfotografia di una RAM da 4.096 bit. *Il chip è di 0,225 × 0,142 pollici (Intel Corp.)*

Una caratteristica importante delle RAM a semiconduttore è il fatto che esse sono *volatili:* perdono cioè i dati in esse memorizzati quando non è presente l'alimentazione, cosicché quando sono riaccese, vengono a contenere dei dati casuali. Nel caso delle ROM tale problema invece non si pone, per cui tali dispositivi vengono utilizzati per memorizzare in modo permanente dati e programmi. Viceversa, poiché non è possibile alterare il contenuto di un ROM, per la memorizzazione temporanea di dati e di programmi, devono essere utilizzate le RAM.

La Figura 1-10 mostra una ROM contenente 2.048 parole di otto bit ciascuna, per un totale di 16.384 bit. Quando si devono usare dei numeri molto grossi che sono delle potenze di due, ci si serve spesso del simbolo «K», ad indicare 1.024 (2^{10}). La memoria, presentata in figura, contiene così 2K byte, ovvero 16K bit. Poiché ogni locazione contiene otto bit, si dice anche che si ha una ROM da 2K × 8.

Quando l'ingresso di selezione del chip ($\overline{CS}$) è basso, i driver d'uscita della ROM sono abilitati. Quando, viceversa, $\overline{CS}$ è alto, le uscite dei dati sono nello stato ad alta impendenza. Il fatto di avere delle uscite a tre stati permette di poter collegare insieme le linee dei dati di più dispositivi, un solo dispositivo per volta essendo selezionato (una sola linea di selezione bassa).

La Figura 1-11 presenta una RAM da 1K × 8bit. La RAM contiene 1.024 locazioni di otto bit ciascuna. Le linee dei dati sono bidirezionali, poiché i dati possono entrare od uscire dalla memoria. Le RAM dispongono di una ulteriore linea di controllo detta $\overline{WRITE}$ (scrittura). Quando si vogliono immagazzinare dei dati nella RAM, viene selezionato un indirizzo, sulle linee dei dati viene posto il dato da memorizzare e la linea $\overline{WRITE}$ è portata bassa. Quando sia il dato che gli indirizzi sono pronti e stabili, viene generato un impulso sulla linea di selezione del chip ed il dato è immagazzinato nella memoria.

La linea di scrittura definisce la direzione di trasferimento del dato. Di solito questa linea è attiva bassa ed è spesso indicata con il nome RD / $\overline{WR}$ (oppure R / $\overline{W}$). Questa notazione significa che viene effettuata una lettura quando tale segnale è alto, mentre si verifica una scrittura quando è basso. Notate che tale ingresso non ha alcun effetto se il segnale di selezione del chip non è vero.

La ROM e le RAM sono disponibili in diverse dimensioni (diverse capacità di parole e diverse lunghezze, cioè numeri di bit, di parola), e in diversi modelli. Nella Lezione 9 saranno introdotte ulteriori nozioni sulle ROM e sulle RAM.

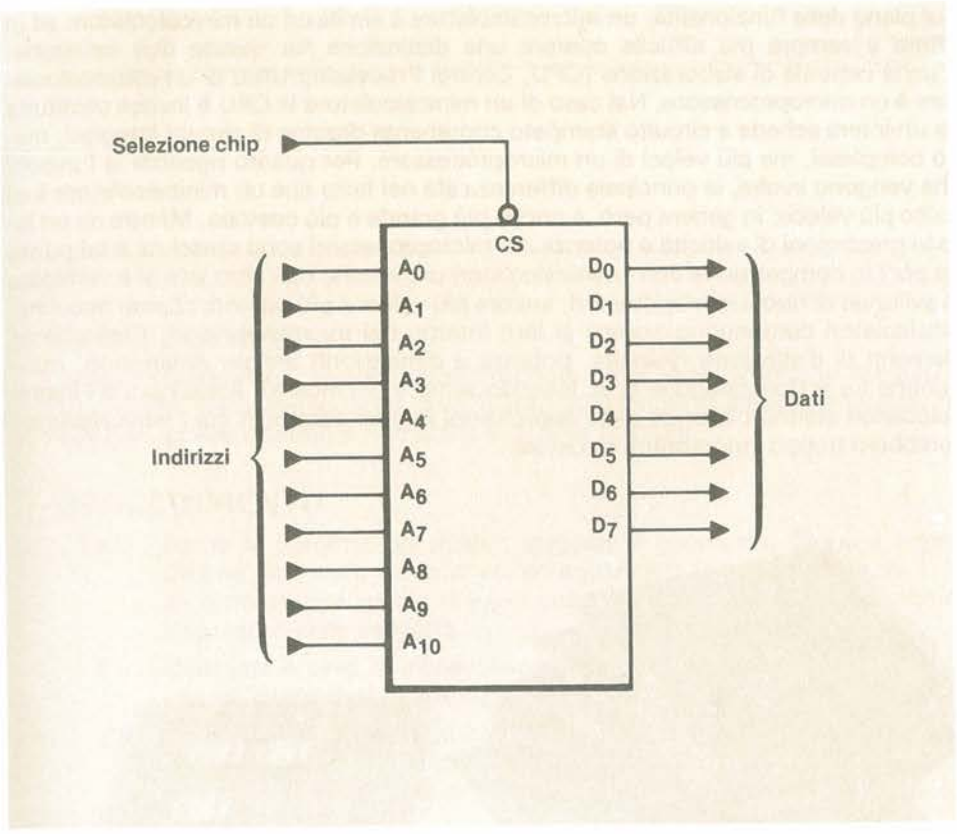


Figura 1-10. ROM da 2K × 8

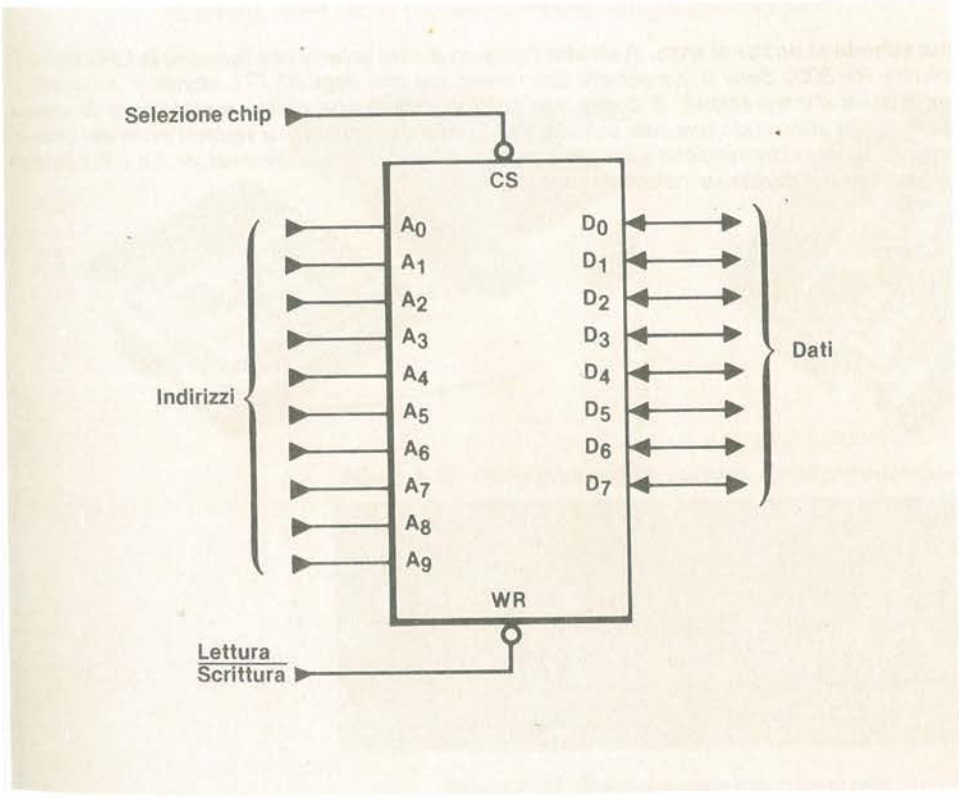
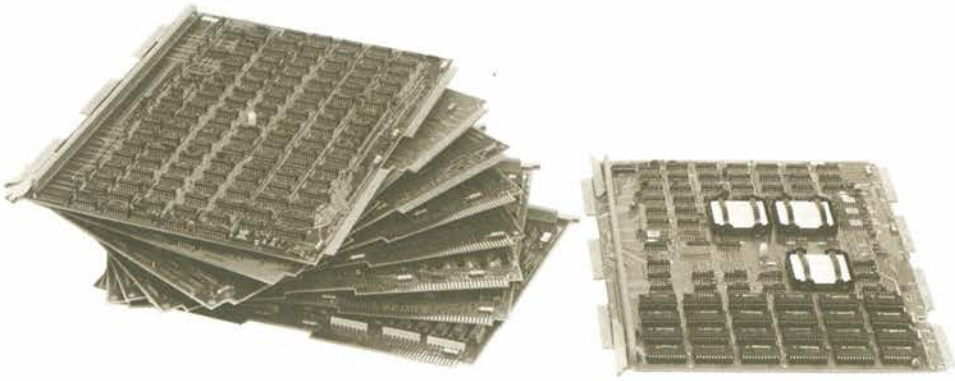


Figura 1-11. RAM da 1K × 8

MICROCALCOLATORI E MINICALCOLATORI

Sul piano della funzionalità, un microcalcolatore è simile ad un minicalcolatore ed in effetti è sempre più difficile operare una distinzione tra queste due categorie. L'unità centrale di elaborazione (CPU, Central Processing Unit) di un microcalcolatore è un microprocessore. Nel caso di un minicalcolatore la CPU è invece costituita da un intera scheda a circuito stampato contenente dozzine di circuiti integrati, meno complessi, ma più veloci di un microprocessore. Per quanto riguarda le funzioni che vengono svolte, la principale differenza sta nel fatto che un minicalcolatore è di solito più veloce: in genere però, è anche più grande e più costoso. Mentre da un lato le prestazioni di velocità e potenza dei microprocessori sono cresciute a tal punto da porli in competizione con i minicalcolatori più vecchi, dall'altro lato si è verificato lo sviluppo di nuovi minicalcolatori, ancora più veloci e più potenti. Questi nuovi minicalcolatori contengono spesso al loro interno dei microprocessori. Cosicché gli elementi di distinzione (velocità, potenza e dimensioni) ancora rimangono, ma il confine tra le due categorie si va facendo sempre più incerto. Attualmente i microcalcolatori stanno trovando delle applicazioni in quei sistemi in cui i minicalcolatori sarebbero troppo ingombranti e costosi.



Una scheda al posto di otto. *A sinistra il gruppo di otto schedė che formano la CPU del calcolatore HP 3000 Serie II. Le schede sono realizzate con degli IC TTL standard a piccola e media scala d'integrazione. A destra una CPU completa che svolge esattamente la stessa funzione ma utilizzando una sola scheda. I tre grossi quadrati che si vedono sono dei circuiti integrati custom che rendono appunto possibile tale riduzione di dimensioni. La CPU su singola scheda è utilizzata sui calcolatori della Serie 33.*

Introduzione al Microprocessor Lab

INTRODUZIONE

Questo esperimento vi serve come introduzione al μLab. Vengono eseguiti numerosi programmi dimostrativi che sono memorizzati nella ROM. Utilizzando lo stesso hardware, ma dei programmi differenti possono essere effettuate numerose e diverse funzioni.

PROCEDIMENTO

A) Aprite la valigetta del μLab e staccate il coperchio. Girate il coperchio in modo che la parte esterna sia rivolta verso di voi ed agganciate le cerniere (Figura 1-12). Piegate il tutto in modo da formare una specie di «A»; collegate a questo punto la cinghietta ai due bottoni posti sul lato destro della valigetta.

B) Collegate il cavo di alimentazione alla rete. Se il μLab non è mai stato usato, verificate prima che gli interruttori di selezione della tensione siano disposti in modo opportuno (Figura 1-13).

C) Accendete il μLab portando l'interruttore di rete sulla posizione ON. Il display e i LED di uscita vengono accesi per circa un secondo e l'altoparlante emette un «bip». Questo sta ad indicare che la verifica automatica, che viene fatta all'accensione, è stata completata in modo corretto. Il display visualizza uLAb UP indicando così che il sistema è pronto e sta aspettando un comando. In qualunque momento vi accada di premere un tasto sbagliato e vogliate riportarvi nello stato iniziale, premete semplicemente RESET.

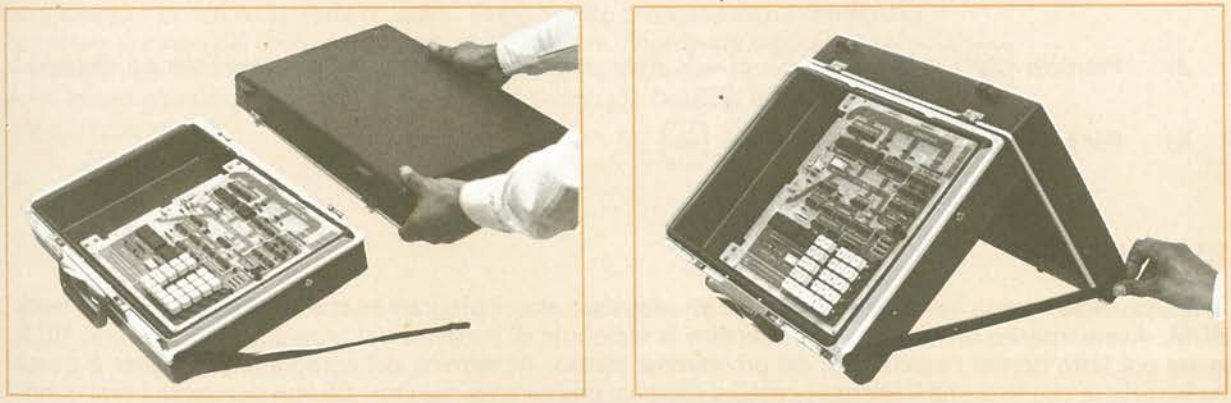


Figura 1-12. Come preparare la valigetta del Microprocessor Lab

Figura 1-13. Selezione della tensione di rete

ESPERIMENTO 1-1

(continuazione)

D) Premete [FETCH ADRS]. Le lineette sul display stanno ad indicare che il μLab sta aspettando che voi inseriate un indirizzo.

E) Premete [0] [5] [F] [9]. Tale numero dovrebbe ora comparire nelle quattro cifre di sinistra del display.

F) Premete [RUN], avete così fatto partire il programma "decollo missile" che è memorizzato nella ROM del μLab.

G) Vedrete a questo punto che alcune luci lampeggiano, e sentirete dei rumori. Il programma dopo un certo tempo si fermerà. Volendo farlo ripartire vi basta premere [RUN].

H) Premete [FETCH ADRS] [0] [5] [3] [E] [RUN]. In questo caso si ha un programma che genera una nota «a caso».

I) Per fermarlo premete [RESET].

J) Premete [FETCH ADRS] [0] [5] [5] [A] [RUN]. Tale programma genera una scritta mobile sul display.

K) Per fermare il programma premete [RESET].

RIASSUNTO

In questo esperimento avete fatto sì che il μLab eseguisse alcuni programmi che sono immagazzinati nella ROM. Avete inserito un indirizzo che specifica la locazione di partenza del programma. Premendo RUN avete poi fatto partire l'esecuzione del programma stesso. Al termine del «programma missile» è posta un'istruzione che fa sí che il microprocessore ritorni al programma monitor. Gli altri due programmi continuano invece a girare, finché non premete RESET. Questi programmi dovrebbero mostrarvi quante funzioni differenti possono essere effettuate utilizzando lo stesso hardware con programmi diversi.

Questi programmi sono costituiti da una lunga serie di eventi, ciascuno dei quali è abbastanza complesso. Grazie al controllo dato dal software, il microprocessore si adatta bene a svolgere compiti di questo tipo. Supponete, invece, di avere un circuito a logica sparsa in grado di effettuare il «decollo missili». Per poter generare una così lunga sequenza di eventi (tra cui il visualizzatore dei numeri e il generatore dei suoni) tale circuito dovrebbe sicuramente essere assai complicato. Nel caso si volesse poi effettuare l'operazione di generare delle note a caso, sarebbe probabilmente necessario rimaneggiare largamente tale circuito. Il microprocessore, al contrario, vi permette di svolgere tutti questi compiti basandosi su un hardware non specializzato, cioè general-purpose. Facendo riferimento alla Appendice E, provate a far girare degli altri programmi dimostrativi. Avete a disposizione un programma «organo», un «cronometro» e un «gioco».

Lezione 1

I sistemi a microprocessore sono dei piccoli calcolatori in grado di scambiare informazioni con dei dispositivi esterni. Tali sistemi dispongono di memoria in cui immagazzinare dati e programmi e di porte d'ingresso e di uscita per comunicare con altri dispositivi. Il nucleo del sistema è detto microcalcolatore, mentre i dispositivi collegati alle porte di I / O sono detti periferiche. All'interno del sistema i componenti sono collegati attraverso dei bus: un bus degli indirizzi, un bus dei dati e un bus di controllo.

Le ROM sono delle memorie che possono essere programmate una volta sola e che non possono più essere modificate. Sono usate per memorizzare in modo permanente i programmi e i dati. Le RAM sono delle memorie in cui è possibile modificare i dati e sono perciò utilizzate per memorizzare, in modo temporaneo, i dati e i programmi. Tuttavia quando vien tolta l'alimentazione la RAM perde l'informazione memorizzata.

I circuiti del microcalcolatore costituiscono l'hardware del calcolatore. Le istruzioni, che formano i programmi e che sono immagazzinate nella memoria, costituiscono il software del calcolatore. Quando il software è memorizzato in modo permanente nella ROM, viene anche indicato con il nome di firmware.

Il microprocessore è solamente in grado di leggere delle istruzioni della memoria e di eseguirle. La possibilità di svolgere i più svariati compiti dipende dal fatto che possono essere eseguite infinite sequenze diverse di istruzione. La gran parte delle applicazioni di un microcalcolatore sono assimilabili a processi costituiti da letture di dati delle porte d'ingresso, manipolazioni degli stessi, realizzazione di un certo numero di scelte e scritture di dati nelle porte d'uscita. Tutto questo processo viene controllato dal software. È molto più facile modificare il software che l'hardware e grazie al software è anche possibile effettuare delle operazioni assai complesse. Grazie a tutto ciò i sistemi basati su microprocessori si dimostrano molto più flessibili e sofisticati dei sistemi a logica sparsa.

DOMANDE

Lezione 1

Nota: le risposte alle domande sono date nell'Appendice A.

1. I sistemi basati su microprocessori sono più flessibili dei sistemi a «logica cablata» o sparsa, in quanto:
   a. sono più veloci.
   b. fanno uso di dispositivi LSI.
   c. la loro funzionalità è controllata da software.
   d. l'hardware è specializzato.

2. La «personalità» di un sistema basato su microprocessore è legata soprattutto a:
   a. il particolare tipo di microprocessore utilizzato.
   b. le periferiche e il software.
   c. il numero di linee di cui è formato il bus dei dati.
   d. il tipo di circuiti integrati di memoria che sono utilizzati.

3. In un sistema a microprocessore, per collegare i componenti si utilizzano tre gruppi di segnali:

   il bus _____, il bus _____, e il bus _____

4. I microprocessori ad otto bit trattano i dati per gruppi di bit che sono detti _____

5. Un sistema a microprocessore comunica con le periferiche attraverso:
   a. le memorie a sola lettura.
   b. le porte d'ingresso e di uscita.
   c. il bus degli indirizzi.
   d. una tastiera.

6. Le periferiche sono:
   a. dispositivi di memoria.
   b. il microprocessore.
   c. il software.
   d. i dispositivi di I / O.

7. I driver a tre stati sono utili in quanto:
   a. permettono a più dispositivi di essere facilmente collegati insieme.
   b. hanno un basso consumo di corrente.
   c. costano poco.
   d. permettono di avere tra livelli logici distinti.

8. Le ROM sono utilizzate soprattutto per:
   a. memorizzare dei dati.
   b. memorizzare dati e programmi in modo temporaneo.
   c. rendere più difficili delle modifiche.
   d. memorizzare dati e programmi in modo permanente.

9. Di solito, per memorizzare informazioni per lunghi periodi di tempo, non si utilizzano le RAM, perché:
   a. perdono il loro contenuto quando viene tolta l'alimentazione.
   b. sono lente.
   c. sono troppo costose.
   d. possono essere scritte una sola volta.

LEZIONE 2

Sistema di numerazione

INTRODUZIONE

In questa lezione verranno discussi i sistemi di numerazione decimale, binaria, ottale ed esadecimale. Noi tutti siamo naturalmente portati a lavorare in decimale, i calcolatori invece operano in binario. Per i numeri binari, sono state introdotte le rappresentazioni esadecimali ed ottale, in quanto più facilmente utilizzabili dalle persone. Il Microprocessor Lab fa uso della notazione esadecimale per rappresentare dati e programmi binari.

DECIMALI E BINARI

Il sistema di numerazione con cui noi tutti abbiamo familiarità, è il sistema *decimale,* quello cioè in base dieci. L'espressione «in base 10» indica che una cifra può servire a rappresentare 10 numeri diversi: 0,1,2,3,4,5,6,7,8 e 9.

I calcolatori fanno invece uso del sistema di numerazione *binario,* ovvero in base due. Con una cifra possono essere rappresentati due soli numeri differenti: 0 e 1. Nel caso dei sistemi digitali è questo un sistema di numerazione del tutto naturale, praticamente obbligato, visto che un segnale digitale può avere due soli stati: basso e alto, che possiamo corrispondentemente indicare come 0 e 1.

La Figura 2-1 confronta il sistema di numerazione decimale e quello binario. Nel sistema decimale le posizioni delle cifre hanno pesi di $10^0=1$, $10^1=10$, $10^2=100$, $10^3=1000$, ecc.
Nel sistema binario si hanno invece pesi di $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, ecc.

DECIMALE

Peso di ciascuna cifra	10^4	10^3	10^2	10^1	10^0
Esempio	1	9	6	0	1

$$19.601 = 1x10^4 + 9x10^3 + 6x10^2 + 0x10^1 + 1x10^0$$
$$= 10.000 + 9.000 + 600 + 1$$

$10^0 = 1$
$10^1 = 10$
$10^2 = 100$
$10^3 = 1.000$
$10^4 = 10.000$

BINARIA

2^4	2^3	2^2	2^1	2^0
1	0	1	1	0

$$10110 = 1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0$$
$$= 16 + 4 + 2$$

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$

Figura 2-1. Sistemi di numerazione binaria e decimale

Per effettuare una conversione da binario a decimale, è sufficiente sommare tutti i valori decimali delle diverse colonne. Così, ad esempio:
$11001 = (1 \times 2^0) + (0 \times 2^1) + (0 \times 2^2) + (1 \times 2^3) + (1 \times 2^4) = 1 + 0 + 0 + 8 + 16 = 25$.
Nella Tabella 2-1 sono riportati gli equivalenti binari dei numeri decimali da 0 a 31 e sono indicate anche le corrispondenti rappresentazioni ottali ed esadecimali.

Binario	Decimale	Ottale	Esadecimale
00000	0	0	0
00001	1	1	1
00010	2	2	2
00011	3	3	3
00100	4	4	4
00101	5	5	5
00110	6	6	6
00111	7	7	7
01000	8	10	8
01001	9	11	9
01010	10	12	A
01011	11	13	B
01100	12	14	C
01101	13	15	D
01110	14	16	E
01111	15	17	F
10000	16	20	10
10001	17	21	11
10010	18	22	12
10011	19	23	13
10100	20	24	14
10101	21	25	15
10110	22	26	16
10111	23	27	17
11000	24	30	18
11001	25	31	19
11010	26	32	1A
11011	27	33	1B
11100	28	34	1C
11101	29	35	1D
11110	30	36	1E
11111	31	37	1F

Tabella 2-1. Sistemi di numerazione

La Figura 2-2 vi illustra una tecnica che vi permette di effettuare una conversione da decimale a binario. Per prima cosa, si divide il numero decimale per due. Il resto di tale divisione, che nel nostro esempio vale 1, viene ad essere il bit meno significativo, quello cioè posto più a destra, del risultato. Il risultato della divisione (nel nostro esempio è 6) è ancora diviso per due. Il resto è il secondo bit del risultato. Questa operazione continua: il resto di ogni divisione contribuisce sempre ad aggiungere un bit al numero binario. Il processo termina quando il risultato della divisione è 0.

OTTALI

Un problema, che nasce quando si utilizzano numeri binari, è quello che essi sono formati da più cifre di quelle che si hanno nel loro equivalente decimale, e di conseguenza sono più difficili da trattare. Ad esempio, dovendo trascrivere il numero «10111001», è abbastanza facile commettere un errore e scrivere così «10110001».

Un modo per ridurre questo problema è quello di utilizzare, per i numeri binari, una rappresentazione più compatta. Si potrebbe usare una rappresentazione decimale, ma la conversione decimale - binario non è sicuramente un'operazione "pulita" ed immediata. La Figura 2-3 vi presenta una rappresentazione detta *ottale*, cioè in base 8. Il numero binario è diviso, a partire da destra, in gruppi di tre bit.

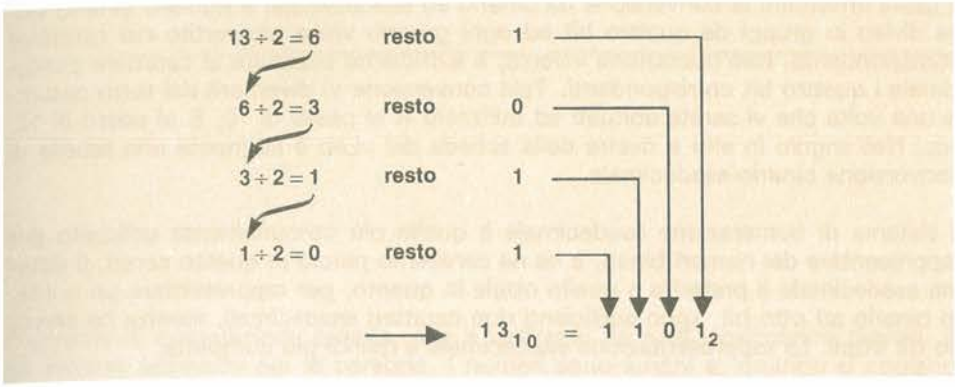


Figura 2-2. Conversione da decimale a binario

Ogni gruppo di tre è sostituito pari pari dal suo equivalente ottale. In questo modo il numero binario 101 001 può quindi essere rappresentato dal numero ottale 51. Notate che nonostante sembrino in tutto e per tutto uguali, i numeri ottali sono diversi dai numeri decimali. Così, l'equivalente decimale di 101001 vale $2^0+2^3+2^5=1+8+32=41$. È molto più facile convertire da binario ad ottale e viceversa che effettuare la conversione binario - decimale.

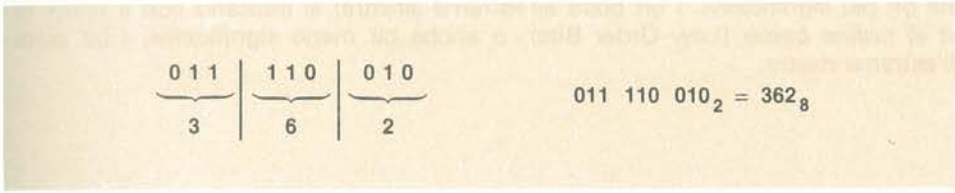
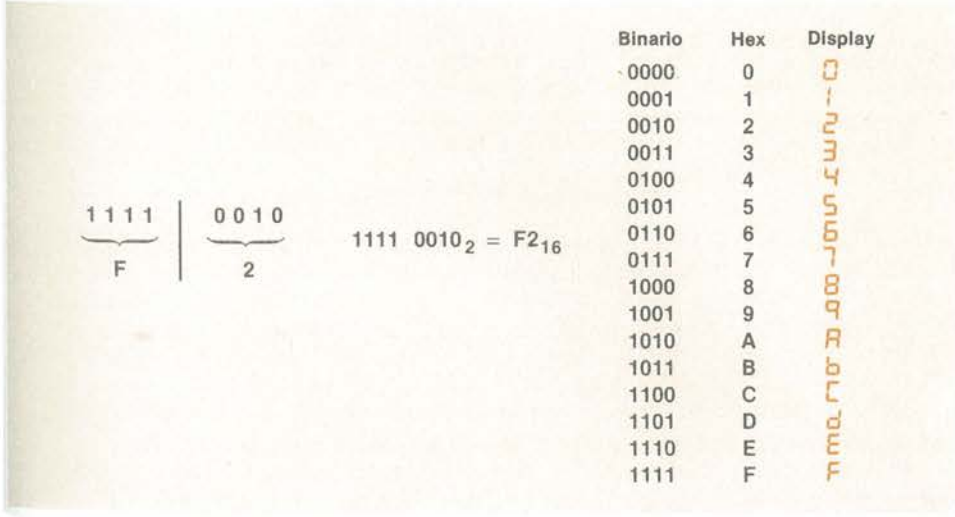


Figura 2-3. Rappresentazione ottale dei numeri

ESADECIMALE

Per i numeri binari un'altra rappresentazione comoda è quella *esadecimale* (Hexadecimal, spesso abbreviato in "Hex"), che si basa cioè su un sistema di numerazione in base 16. Ogni gruppo di quattro bit viene sostituito da un carattere (riferitevi alla Figura 2-4). Poiché quattro bit possono rappresentare dei valori decimali da 0 a 15, è utile trovare un modo per rappresentare con un solo carattere i valori decimali da 10 a 15. A tal fine vengono utilizzati le lettere dalla A alla F. In esadecimale quindi conterete (come è mostrato nella Tabella 2-1) 0,1,2 ...8,9,A,B,C,D,E,F,10,11...



Binario	Hex	Display
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	A
1011	B	b
1100	C	C
1101	D	d
1110	E	E
1111	F	F

Figura 2-4. Rappresentazione esadecimale (Hex) dei numeri

È facile effettuare la conversione da binario ad esadecimale. Il numero binario viene diviso in gruppi da quattro bit ed ogni gruppo viene convertito nel carattere corrispondente. Nell'operazione inversa, è sufficiente sostituire al carattere esadecimale i quattro bit corrispondenti. Tale conversione vi diventerà del tutto naturale una volta che vi sarete abituati ad utilizzare A al posto di 10, B al posto di 11, ecc. Nell'angolo in alto a destra della scheda del μLab è stampata una tabella di conversione binario-esadecimale.

Il sistema di numerazione esadecimale è quello più comunemente utilizzato per rappresentare dei numeri binari, e ce ne serviremo perciò in questo corso. Il sistema esadecimale è preferito a quello ottale in quanto, per rappresentare un numero binario ad otto bit, sono sufficienti due caratteri esadecimali, mentre ne servono tre ottali. La rappresentazione esadecimale è quindi più compatta.

NOMENCLATURA DEI BIT

Quando si hanno dei numeri binari, è spesso necessario fare riferimento ad un bit o a dei gruppi di bit particolari. Il bit posto all'estrema destra è detto *bit meno significativo* (Least Significant Bit, LSB) mentre il bit posto alla estrema sinistra è detto *bit più significativo* (Most Significant Bit, MSB).

Quando ci si vuole riferire a un gruppo di bit posto ad una delle due estremità di una parola, si indicano con il nome di *bit di ordine alto* (High-Order Bits), o anche bit più significativi, i bit posti all'estrema sinistra; si indicano con il nome di *bit di ordine basso* (Low-Order Bits), o anche bit meno significativi, i bit posti all'estrema destra.

Lezione 2

Il sistema di numerazione binaria, che è utilizzato da tutti i calcolatori digitali, è un sistema scomodo per le persone. I numeri sono lunghi e, quando si copiano dei numeri, è facile fare degli errori. Per le persone il sistema più naturale è quello decimale: è però difficile la conversione decimale-binario. I sistemi di numerazione esadecimale ed ottale hanno delle rappresentazioni simili a quella decimale, ma permettono di effettuare delle conversioni in binario in modo molto più semplice. In questo corso è usato il sistema esadecimale (Hex).

In questa lezione sono stati considerati solo numeri interi positivi. Verranno descritti invece nella lezione 15 dei modi per rappresentare i numeri negativi, le frazioni e i numeri molto grossi e molto piccoli.

DOMANDE

Lezione 2

1. Una cifra binaria è un _____ o uno _____ .

2. L'equivalente decimale di 10101100 è _____ .

3. L'equivalente ottale di 10101100 è _____ .

4. L'equivalente esadecimale di 10101100 è _____ .

5. L'equivalente binario di B9 (hex) è _____ .

6. 11111111 (binario) = 255 (decimale) = _____ (hex) = _____ (ottale).

7. Quando si lavora con numeri binari è preferibile utilizzare la rappresentazione esadecimale invece che quella decimale perché:

   a. Il sistema esadecimale è più facilmente utilizzabile dalle persone.
   b. Il sistema esadecimale dà una rappresentazione meno compatta.
   c. I numeri esadecimali sono legati ai numeri binari da relazioni più semplici.
   d. I numeri decimali sono legati ai numeri binari da relazioni più semplici.

LEZIONE 3

Elementi di software

INTRODUZIONE

Come già accennato nelle precedenti lezioni, un sistema microcalcolatore è composto sia di hardware che di software. In questa lezione descriveremo alcuni aspetti software (ovvero di programmazione) del sistema.

In un primo tempo i programmi vengono scritti in un linguaggio comodo per la persona che li scrive (il *programmatore*). Successivamente i programmi devono essere riscritti, e immagazzinati in un linguaggio comprensibile al microprocessore. Il microprocessore preleva quindi tali codici dalla memoria uno per volta; in modo sequenziale, ed esegue le operazioni in essi indicate.

È importante capire quello che può e quello che non può essere fatto da un microcalcolatore. Di tali limiti parleremo appunto nella prima parte di questa lezione.

I CALCOLATORI NON PENSANO

Quando si scrive un programma per un calcolatore è necessario spiegare al calcolatore tutto quello che deve fare, non solo a grandi linee ma nei più minuti particolari. In effetti, se paragonati agli uomini, i calcolatori non si dimostrano eccessivamente intelligenti (si veda in tal senso la Figura 3-1). Una lampadina è del tutto priva di intelligenza; basta girare l'interruttore e si accende. Un tostapane è già un po' meglio, visto che può darci un toast cotto al punto giusto. Tra le macchine comunque il più alto grado di intelligenza è raggiunto dai calcolatori che sono un po' l'ultimo grido nel campo della intelligenza meccanica. E tuttavia un semplice lombrico è più intelligente di un calcolatore.

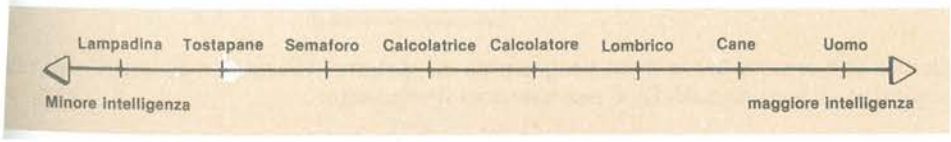


Figura 3-1. Livelli di intelligenza

È a questo punto importante distinguere tra due concetti: quello di intelligenza e quello di capacità di calcolo. I calcolatori sono in grado di eseguire operazioni ad una velocità e con una precisione eccezionali; è però necessario indicare loro esattamente quello che deve essere eseguito. Sono in grado di dare risposte corrette anche di fronte al verificarsi di mutamenti di certe condizioni, basta che contengano un programma che dica loro: «Quando si verifica questa condizione esegui questa certa cosa». Un lombrico viceversa non agisce seguendo una sequenza di istruzioni ma secondo una sua strategia generale, all'interno della quale affronta le nuove situazioni createsi. Anche se in misura limitata, mostra di possedere quelle misteriose caratteristiche che sono proprie della creatività e della possibilità di apprendere. All'opposto i calcolatori operano solo in modo logico e prefissato.

È importante quindi evidenziare che i calcolatori sono veloci e precisi, ma non creativi. Sono in grado di trattare situazioni diverse e mutevoli, ma unicamente nei modi e nei limiti dei programmi in essi contenuti: la loro apparente intelligenza è in realtà dovuta solo a tali programmi. Con l'uso dei microprocessori è diventato possibile inserire in diversi prodotti un po' di questa «finta» intelligenza, ottenendo un livello di sofisticazione che sarebbe viceversa assai difficile ottenere.

IL MICROPROCESSORE NELLE VESTI DI UN DISPOSITIVO LOGICO

I sistemi a microprocessore sono spesso utilizzati per sostituire circuiti realizzati con componenti logici standard. Al fine di illustrare le differenze che esistono tra dispositivi logici programmabili e dispositivi logici tradizionali, possiamo pensare di utilizzare un microprocessore come se fosse un dispositivo AND qualunque.

Per realizzare un AND con un microprocessore, saranno necessarie una porta d'ingresso, corrispondente alle linee di ingresso del dispositivo AND e una porta d'uscita per le sue linee d'uscita (Figura 3-2). La funzione logica AND sarà effettuata dal microprocessore grazie alle istruzioni presenti in memoria. In realtà, poiché un AND ha una sola uscita, della porta d'uscita sarà necessario utilizzare un solo bit.

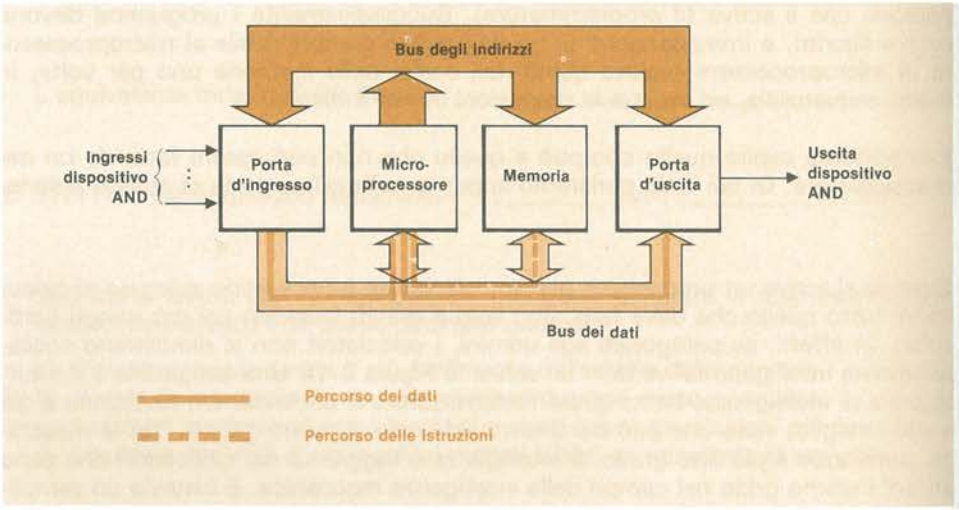


Figura 3-2. Dispositivo AND realizzato con un microprocessore

Quello che ancora serve è un programma opportuno. Un elenco di istruzioni, che realizzino la funzione AND, è per esempio il seguente:

1. Leggi la porta d'ingresso.
2. Salta al punto 5 se tutti gli ingressi sono alti; altrimenti procedi.
3. Porta bassa l'uscita.
4. Salta al passo 1.
5. Porta alta l'uscita.
6. Salta al passo 1.

Dapprima viene letta la porta d'ingresso. Quindi sono controllati gli ingressi per vedere se sono tutti alti, esattamente la funzione di un AND. Se tutti gli ingressi sono alti, l'uscita è portata alta; in caso contrario è portata bassa. Quando è terminata questa sequenza, il programma ritorna al passo 1 e continua a ripetere la sequenza di istruzioni cosicché l'uscita seguirà le variazioni degli ingressi.

DIAGRAMMI DI FLUSSO

I *diagrammi di flusso* (flowchart) sono un metodo grafico per descrivere le operazioni effettuate da un programma e sono costituiti da diversi tipi di blocchi collegati da linee. In Figura 3-3 sono mostrati i tre principali tipi di blocchi utilizzati nei diagrammi di flusso. Un blocco di forma rettangolare serve a descrivere le azioni effettuate dal programma. Un blocco di forma romboidale è invece utilizzato per indicare una decisione, per esempio un test sul valore di una variabile. Infine con un ovale si è soliti iniziare un programma, il nome dello stesso essendo riportato all'interno dell'ovale. Lo stesso simbolo può essere utilizzato anche per chiudere un diagramma di flusso. Esistono poi numerosi altri simboli particolari, ma di essi non faremo uso in questo corso.

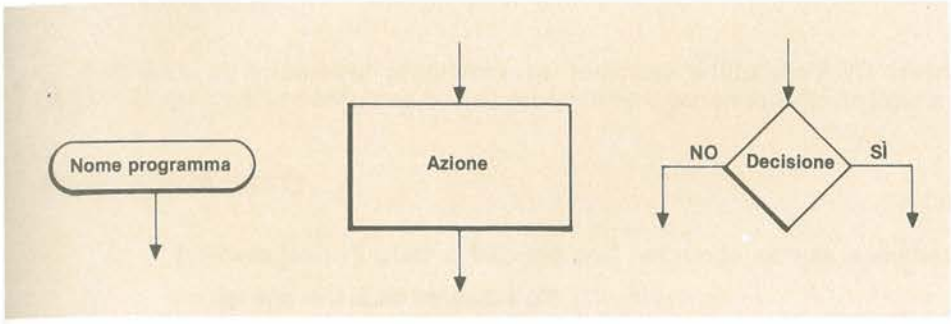


Figura 3-3. Simboli dei diagrammi di flusso

La Figura 3-4 presenta un diagramma di flusso di un dispositivo AND. Ad ogni linea del programma corrisponde un blocco: fanno eccezione le istruzioni di salto che sono rappresentate solo da una linea. Le linee complessivamente mostrano il percorso, il flusso cioè del programma da un blocco ad un altro.

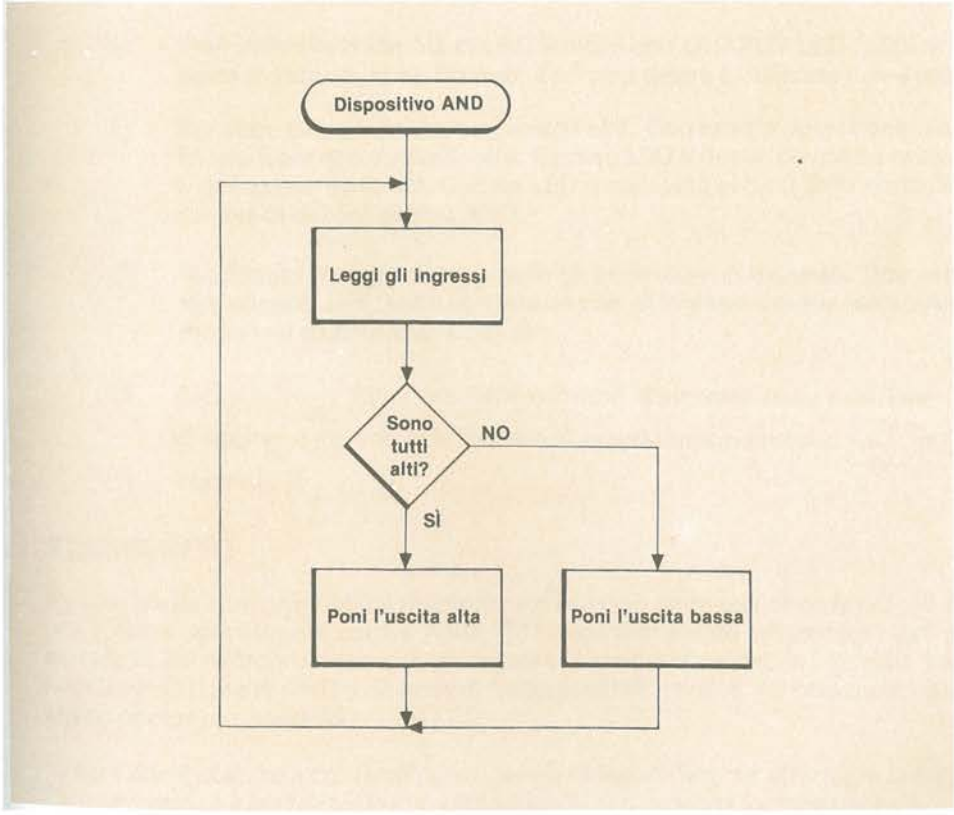


Figura 3-4. Diagramma di flusso del dispositivo AND

Se è vero che nel diagramma di flusso sono presenti le stesse informazioni che sono contenute nell'elenco di istruzioni già visto, esse sono però mostrate in forma grafica: quando si inizia a scrivere un programma, un buon metodo per organizzare le idee e per documentare quali operazioni devono essere svolte dal programma, è quello di servirsi di tali diagrammi. Ripercorrendo passo passo il diagramma di flusso se ne può fare una verifica logica a tavolino. Successivamente, dal diagramma di flusso si può procedere a scrivere il programma. Ancora, i diagrammi di flusso sono utili per riguardare e comprendere un programma che sia stato scritto qualche tempo prima.

ESPERIMENTO 3-1

Il dispositivo AND col Microprocessor Lab

INTRODUZIONE

La ROM del μLab contiene un programma per realizzare la funzione AND descritta. In questo esperimento vi verrà insegnato come far girare questo programma e verranno indicate alcune osservazioni significative.

PROCEDIMENTO

A) Premete [RESET]. Il μLab si dispone così nel modo «attesa comando». Tale operazione non è necessaria se il μLab visualizza già uLAb UP.

B) Premete [FETCH REG] [0] [4] [E] [0]. Con questo viene specificato l'indirizzo di partenza del programma.

C) Premete [RUN]. Sta ora girando il programma «AND GATE» (il display è abbuiato).

D) Identificate sulla scheda in basso gli otto interruttori indicati con INPUT. Questi interruttori sono collegati alla porta d'ingresso e corrispondono agli ingressi del dispositivo AND.

E) Fate attenzione alla fila di LED indicati con OUTPUT LED (LED di uscita). Sono collegati alla porta d'uscita e, in particolare, il primo a destra è utilizzato come uscita del dispositivo AND.

F) Spostare tutti gli interruttori verso l'*alto*. Con questa operazione, su ogni linea della porta d'ingresso è portato un livello alto. Il primo LED a destra dovrebbe essere ora acceso, dato che tutti gli ingressi sono alti. Questo LED è collegato al bit 0 della porta di uscita, che è utilizzata come uscita del dispositivo AND.

G) Modificate gli ingressi attraverso gli interruttori di ingresso. Dovreste notare che il LED di uscita si spegne, indicando con questo che gli ingressi del dispositivo AND non sono più disposti in modo tale da dare una uscita alta.

H) Premete [RESET]. Riportate gli interruttori d'ingresso nella posizione iniziale. Notate che il LED di uscita, dato che si è fermato il programma premendo [RESET], non segue più lo stato degli ingressi.

RIASSUNTO

Il μLab, grazie a un programma memorizzato in modo permanente nella propria ROM, è stato così utilizzato per operare come un dispositivo AND. Gli interruttori hanno presentato i dati alla porta d'ingresso, che è stata letta dal microprocessore. Il processore, a seconda dei dati in ingresso, ha quindi deciso se era verificata la condizione di AND e, in base al risultato di tale verifica, ha presentato sulla porta di uscita uno stato logico opportuno, visualizzato dal LED.

Notate che il μLab non contiene alcun circuito specializzato ad effettuare la funzione del dispositivo AND. Tale operazione è invece realizzata utilizzando le sue capacità logiche generali.

CARATTERISTICHE DEL DISPOSITIVO AND REALIZZATO CON UN MICROPROCESSORE

Un sistema a microprocessore può svolgere la stessa funzione di un dispositivo AND; è solo più lento. Anche se a voi può sembrare istantaneo, in realtà ha una velocità molto minore di quella della logica digitale. Il programma per il dispositivo AND è costituito da sei istruzioni, per l'esecuzione di ciascuna delle quali il Lab impiega circa due microsecondi. Possiamo quindi dire che il dispositivo AND realizzato con un microcalcolatore ha un *ritardo di propagazione* (Propagation Delay Time; ovvero il tempo che intercorre tra l'istante in cui si verifica una variazione all'ingresso e l'istante in cui il risultato di tale variazione è riportato all'uscita) pari a circa 2 microsecondi × 6 = 12 microsecondi. Un dispositivo AND standard della serie TTL ha invece un tempo di ritardo pari a circa 10 nanosecondi: mille volte più veloce. Già questo semplice esempio mostra la lentezza di un microprocessore rispetto ai circuiti logici convenzionali. Per sistemi più complessi la differenza diventa ancora maggiore. Tuttavia questo fatto non riveste alcuna importanza in molte applicazioni in cui la velocità della logica tradizionale supera di gran lunga le esigenze specifiche. Prendiamo per esempio una segretaria che batte a macchina, riuscendo a raggiungere la velocità abbastanza inverosimile di cento battute al secondo (più di mille parole al minuto); in questo caso il processore ha comunque tempo 10 millisecondi per rispondere ad ogni tasto battuto, tempo durante il quale il microprocessore può eseguire circa 5000 istruzioni.

Vi potrà stupire il fatto che si sia utilizzato un sistema così complesso quando un semplice dispositivo AND sarebbe stato sufficiente. In effetti, se questa fosse la sola funzione richiesta al sistema, si sarebbe usato proprio un dispositivo AND. Tuttavia il microprocessore è estremamente flessibile e, grazie a tale caratteristica, si possono arbitrariamente ridefinire le funzioni del dispositivo, semplicemente modificando il programma. Si possono aggiungere altri ingressi e la funzione del dispositivo può diventare assai complessa. Prendiamo ad esempio un dispositivo logico ad otto ingressi che funzioni da antifurto elettronico. L'uscita diventa alta solo se gli ingressi vengono modificati seguendo un certo ordine. Mentre utilizzando la logica tradizionale è necessario realizzare un nuovo circuito abbastanza complesso, per tale applicazione può essere utilizzato lo stesso sistema a microprocessore di prima. Naturalmente servirà un nuovo programma, più complicato del semplice programma AND, ma l'hardware (a parte i dispositivi di I / O) non verrà per nulla modificato. Inoltre sarà sufficiente alterare il programma per cambiare la «combinazione» dell'antifurto elettronico.

LINGUAGGI DI PROGRAMMAZIONE

Il fatto di scrivere programmi in italiano può essere molto comodo per noi, in quanto l'italiano è la lingua che usiamo normalmente, ma non è purtroppo altrettanto comodo per un microprocessore, per cui tale linguaggio è del tutto privo di significato. Il linguaggio, che il microprocessore utilizza e quindi comprende, è detto *linguaggio macchina* (Machine Language), spesso chiamato anche *codice macchina* (Machine Code). Poiché i microprocessori trattano direttamente solo dei segnali digitali, le istruzioni espresse in linguaggio macchina sono dei codici binari (P.es., «00111100»). Ancora, il microprocessore è progettato in modo da riconoscere solo un certo, ben definito, gruppo di codici che complessivamente sono detti *insieme di istruzioni* (Instruction Set).
Il linguaggio macchina non è facilmente utilizzabile dalle persone: così «00111100» non ha un significato ovvio. È già più facile lavorare con questo linguaggio se si fa uso di una rappresentazione esadecimale: «0011 1100» può essere sostituito da «3C». Tuttavia anche in questo modo non abbiamo alcuna informazione sul significato di tale istruzione.

Un passo avanti si ha sostituendo il codice macchina di ogni istruzione con un nome abbreviato detto *codice mnemonico* (o semplicemente *mnemonico*). Così il codice «3C», che nel caso del microprocessore 8085 vuol dire «Incrementa il registro A», viene rappresentato dalla dizione «INR A». È molto più facile ricordare i codici mnemonici che ricordare i corrispondenti codici macchina. Assegnando un codice mnemonico ad ogni codice istruzione è possibile scrivere un programma utilizzando mnemonici al posto di codici. Quando è stato scritto tutto il programma, è facile poi convertire i codici mnemonici in codici macchina. Non sarà più necessario che teniate a mente i codici macchina e vi sarà contemporaneamente più facile ricordare il significato delle diverse istruzioni. I programmi scritti utilizzando dei codici mnemonici sono detti programmi in *linguaggio di assemblaggio* (Assembly Language).

Il linguaggio macchina è in genere strettamente legato al progetto del microprocessore stesso, e come tale non può essere modificato. I codici mnemonici del linguaggio di assemblaggio, invece, sono realizzati dal produttore del microprocessore per facilitare il lavoro dei programmatori e non sono vincolati al progetto del processore. Potete così scrivere INC A al posto di INR A: basta che entrambe queste dizioni vengano tradotte nel codice macchina 3C. In questo corso verranno seguite le convenzioni del linguaggio di assemblaggio Intel 8085, che come per la maggior parte dei linguaggi di assemblaggio utilizza delle abbreviazioni di espressioni in inglese.

Se da un lato il linguaggio di assemblaggio rappresenta già un grosso miglioramento rispetto al linguaggio macchina, d'altro canto esso è ancora poco adatto quando si vogliono scrivere dei programmi complessi. Per semplificare l'operazione di programmazione sono stati sviluppati dei *linguaggi ad alto livello* (High-Level Language). Questi linguaggi sono abbastanza simili all'inglese e sono di solito indipendenti dal tipo particolare di microprocessore. Un'istruzione tipica può essere «LET COUNT = 10» oppure «PRINT COUNT».

Queste istruzioni generano dei comandi molto più complicati di quelli che il microprocessore è in grado di comprendere. Per questo motivo i microcalcolatori che utilizzano tali linguaggi ad alto livello contengono di solito dei grossi e complessi programmi (presenti in modo permanente nella loro memoria), che traducono il programma scritto in un linguaggio ad alto livello in un programma scritto in codice macchina.
Una sola istruzione di linguaggio ad alto livello può corrispondere a dozzine di istruzioni in linguaggio macchina. I programmi traduttori sono detti *compilatori* (Compilers).

UN ESEMPIO DI PROGRAMMAZIONE

Per illustrare questi concetti possiamo utilizzare un programma molto semplice. La Figura 3-5 riporta il diagramma di flusso di un programma che conta fino a 10. Questo programma non ha né ingressi né uscite: si limita a far sì che una certa locazione di memoria conti da zero a dieci in modo continuo.

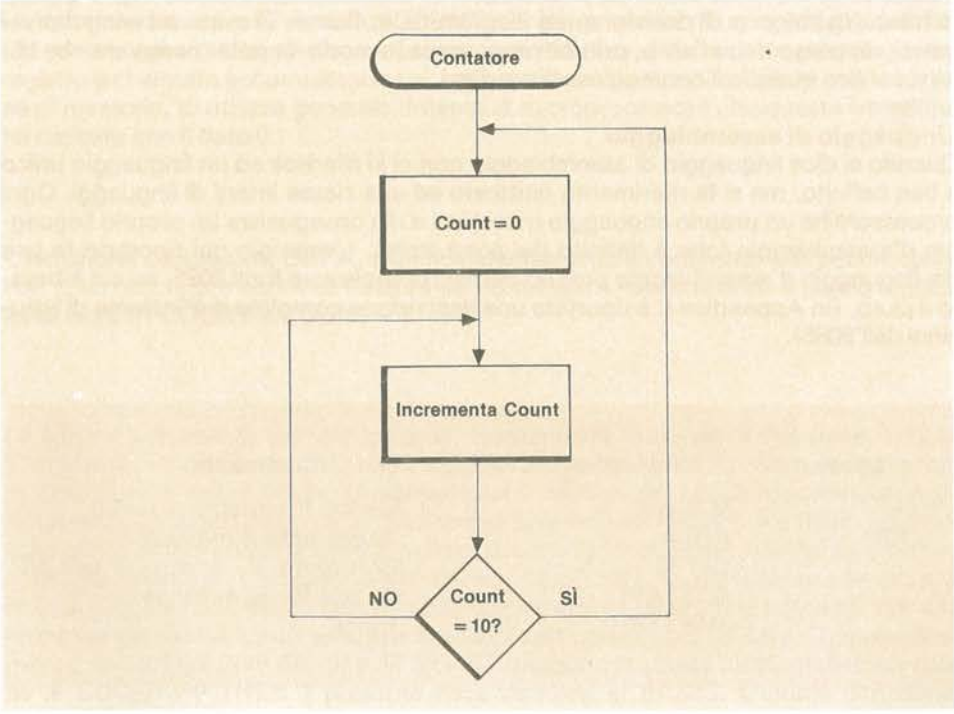


Figura 3-5. Diagramma di flusso del programma «Conta a 10»

I programmi di seguito riportati servono solo a darvi un'idea di come si presentano i diversi tipi di linguaggi. Non preoccupatevi di tenerne a mente i particolari: la programmazione sarà discussa più a fondo nei prossimi capitoli.

Linguaggi ad alto livello

Il passaggio dal diagramma di flusso ad un linguaggio ad alto livello è semplice e diretto. In questo esempio utilizzeremo una variante di quel linguaggio ad alto livello che si chiama *BASIC,* linguaggio che ha il vantaggio di essere semplice e simile alla lingua inglese. Il *listing* (l'elenco cioè delle istruzioni del programma) è il seguente:

N° di linea	Istruzione	Descrizione
1	LET COUNT = 0	Poni COUNT a 0
2	LET COUNT = COUNT + 1	Incrementa COUNT
3	IF COUNT = 10 THEN 1	Vai a 1 se COUNT = 10
4	GO TO 2	Altrimenti vai a 2

Tabella 3-1. Programma «Conta a 10» in linguaggio BASIC

Le prime due linee del programma corrispondono esattamente ai due primi blocchi di azione del diagramma di flusso. Nella prima linea viene caricata, con il valore zero, la locazione di memoria chiamata COUNT (cioè conteggio). La seconda linea, LET COUNT = COUNT + 1, è un modo per dire che la locazione COUNT deve essere incrementata. Le linee tre e quattro realizzano la funzione indicata dal blocco di decisione. La linea tre specifica che, se COUNT = 0, come istruzione successiva deve essere eseguita la linea uno. Se COUNT = 0, tale istruzione non ha invece alcun effetto e il programma può così continuare con la linea quattro. Questa istruzione, a sua volta, indica di andare alla linea due. Queste due istruzioni perciò effettuano quanto richiesto dal blocco di decisione del diagramma di flusso. Provate ad eseguire voi stessi, un passo dopo l'altro, questo programma in modo da poter osservare che cosa si verifica quando il conteggio arriva a dieci.

Linguaggio di assemblaggio

Quando si dice linguaggio di assemblaggio non ci si riferisce ad un linguaggio unico e ben definito, ma si fa riferimento piuttosto ad una classe intera di linguaggi. Ogni processore ha un proprio linguaggio macchina e di conseguenza un proprio linguaggio d'assemblaggio (che è definito dal produttore). L'esempio qui riportato fa uso del linguaggio d'assemblaggio proprio del microprocessore Intel 8085, su cui è basato il μLab. (In Appendice B è riportata una descrizione completa dell'insieme di istruzioni dell'8085).

Label	Istruzione	Commenti
START:	MVI A,0	;Carica il registro A con 0
LOOP:	INR A	;Incrementa il registro A
	CPI 10	;Confronta il registro A con 10
	JZ START	;Se A = 10 vai all'inizio
	JMP LOOP	;Ripeti

Tabella 3-2. Programma «Conta a 10» in linguaggio d'assemblaggio del microprocessore 8085

La tabella 3-2 descrive, in linguaggio d'assemblaggio, il listing del programma «conta a 10». Rispetto a quello scritto in linguaggio BASIC, questo programma è certamente più oscuro; tuttavia entrambi realizzano la stessa funzione. Ricordatevi che le caratteristiche del linguaggio macchina sono strettamente legate alle caratteristiche del microprocessore stesso. Questo programma è perciò diverso da quello in BASIC, che è legato più all'inglese che al linguaggio macchina del microprocessore.

Le tre colonne presenti servono per le *label*, le *istruzioni* ed i *commenti*. La label (etichetta) ha la stessa funzione che avrebbe il numero della linea. Invece di numerare le diverse linee del programma, potete così semplicemente associare un nome (la label appunto) ad ogni linea a cui volete fare riferimento. Per identificare le label si usa il due punti (:). Una linea ha bisogno di una label solo se esiste un'altra istruzione del programma che voglia ad essa riferirsi. La label vi permette di identificare con facilità e chiarezza la linea a cui volete saltare nel corso dell'esecuzione di un programma.

I commenti sono solo un ausilio che vi permette di spiegare e al tempo stesso capire il programma. L'inizio di un commento è identificato da un punto e virgola (;). Nel caso di programmi scritti in linguaggi ad alto livello non si ha bisogno di molti commenti in quanto le istruzioni stesse sono più chiare. Quando si hanno invece programmi scritti in linguaggio di assemblaggio i commenti si rivelano essere un aiuto estremamente positivo. In effetti sono utili sia a coloro che, pur non avendo scritto il programma hanno bisogno di capirlo, sia al programmatore stesso che debba ritornare sul programma dopo che già l'aveva terminato ed abbandonato.

La prima istruzione è MVI A, 0 (Muovi in modo immediato nell'accumulatore il dato zero). L'*accumulatore* (Accumulator, detto anche registro A) è una locazione, interna al microprocessore, in cui possono essere immagazzinati dei dati. Questa istruzione è l'equivalente della LET COUNT = 0, unica differenza il fatto che invece di inventarci un nome per la variante (COUNT), ci siamo serviti di un nome già definito (A) per un registro interno al microprocessore. Più oltre vi verrà detto perché questo registro è chiamato accumulatore; per ora consideratelo semplicemente una locazione di memoria, di utilizzo generale, interna al microprocessore che questa istruzione ha caricato con il dato 0.

L'istruzione successiva, INR A, significa «Incrementa il contenuto dell'accumulatore». L'accumulatore contiene il valore del conteggio e tale istruzione è perciò equivalente alla LET COUNT = COUNT + 1.

Le istruzioni seguenti, nel loro insieme, realizzano la funzione di decisione. CPI 10 (Confronta in modo immediato) ha il significato di «Confronta il contenuto dell'accumulatore con il valore dieci». Qualunque sia il risultato del confronto, non viene direttamente generato alcun salto. Se il valore presente nel registro A è dieci, viene invece posto a uno un particolare flip-flop, detto *flag* (indicatore, bandierina) interno al microprocessore. L'istruzione successiva, JZ START, fa riferimento allo stato di questo flag. Se i due valori erano uguali, tale istruzione vede il flag posto ad uno e fa sì che venga effettuato un salto alla linea indicata con la label START. Complessivamente, le due istruzioni CPI 10 e JZ START svolgono la stessa funzione dell'istruzione IF COUNT = 0 THEN 1 presente nella versione in BASIC. L'ultima istruzione, JMP LOOP, genera semplicemente un salto alla linea indicata dalla label LOOP, ed è così equivalente all'istruzione GO TO 2 in BASIC.

Linguaggio macchina

Come ultima fase del nostro esempio, la tabella 3-3 presenta il listing del linguaggio macchina corrispondente al programma in linguaggio di assemblaggio appena discusso. La funzione del programma è, in questo caso, totalmente nascosta e si mostrano così in tutta la loro evidenza i problemi connessi al fatto di lavorare in linguaggio macchina. Tuttavia l'utilizzo di tale linguaggio è di fatto obbligatorio nella misura in cui è l'unico linguaggio che il microprocessore è in grado di comprendere in modo diretto. Notate che siamo partiti dando del programma una descrizione in italiano e che siamo giunti, alla fine, ad avere una sequenza di zeri e di uno adatta ad essere immagazzinata nella memoria del microprocessore.

Indirizzo di memoria (Hex)	Contenuto della memoria (Hex)	(Binario)
07F0	3E	00111110
07F1	00	00000000
07F2	3C	00111100
07F3	FE	11111110
07F4	0A	00001010
07F5	CA	11001010
07F6	F0	11110000
07F7	07	00000111
07F8	C3	11000011
07F9	F2	11110010
07FA	07	00000111

Tabella 3-3. Listing del programma «Conta a 10» in linguaggio macchina

Per capire il codice macchina fate riferimento alla Tabella 3-4 che confronta le tre versioni del nostro programma. Nel μLab ogni locazione di memoria può contenere otto bit di dato (otto bit possono essere rappresentati con due caratteri esadecimali). Ogni istruzione inizia con un *codice operativo* (Operation Code, spesso abbreviato in opcode). Il codice operativo definisce quale operazione deve essere effettuata. A seconda dell'istruzione, il codice operativo può poi essere seguito da zero, uno o due byte di dato.

Linguaggio BASIC		Linguaggio di assemblaggio, 8085			Linguaggio macchina 8085		
N° Linea	Istruzione	Label	Istruzione		Indirizzo	Codice	
1	LET COUNT = 0	START:	MVI	A,0	07F0	3E	Codice operativo
					07F1	00	Dato
2	LET COUNT = COUNT + 1	LOOP:	INR	A	07F2	3C	Codice operativo
3	IF COUNT = 10 THEN 1		CPI	10_{10}	07F3	FE	Codice operativo
					07F4	0A	Dato
			JZ	START	07F5	CA	Codice operativo
					07F6	F0	Indirizzo
					07F7	07	
4	GO TO 2		JMP	LOOP	07F8	C3	Codice operativo
					07F9	F2	Indirizzo
					07FA	07	

Tabella 3-4. Programma «Conta a 10» nei tre linguaggi

Il primo byte (3E) all'indirizzo 07F0 è il codice operativo dell'istruzione MVI A. È tuttavia solo una parte dell'istruzione. Tale byte indica infatti che voi volete trasferire nell'accumulatore un certo dato; vi serve però anche un'altra locazione di memoria in cui definire tale dato. La successiva locazione di memoria (di indirizzo 07F1) contiene perciò 00, il dato cioè che deve essere caricato nell'accumulatore.

La terza locazione contiene il codice operativo della seconda istruzione, INR A. Questo codice operativo (3C) dice al microprocessore di incrementare l'accumulatore. Poiché nessun dato supplementare è associato ad essa, tale istruzione occupa una sola locazione di memoria.

Il codice operativo FE è il codice operativo dell'istruzione di confronto (Compare) CPI. Esattamente come nel caso della istruzione MVI A, 0, la locazione che segue il codice operativo contiene il dato richiesto dall'istruzione. Poiché il linguaggio macchina è mostrato in notazione esadecimale, il nostro dato (10 in decimale) diventa 0A (in esadecimale). Questa istruzione effettua il confronto tra l'accumulatore e il valore 10 e porta ad uno un flag (come descritto precedentemente) se i due valori sono uguali.

L'istruzione JZ ha il codice operativo CA, che è presente all'indirizzo 07F5. Questo codice operativo dice al microprocessore di effettuare un salto nel caso che il flag sia posto ad uno. Le due locazioni di memoria successive specificano l'indirizzo a cui saltare. Nel sistema 8085 gli indirizzi sono lunghi 16 bit e per memorizzare un indirizzo sono perciò necessarie due locazioni di memoria (di 8 bit ciascuna). Le due parti dell'indirizzo vengono memorizzate nell'ordine opposto a quello che voi vi aspettate. Viene prima memorizzata la metà meno significativa e poi la metà più significativa. In questo modo l'indirizzo 07F0 viene immagazzinato come F0 07. L'istruzione, in linguaggio d'assemblaggio, «JZ START» vuol dire che il processore dovrebbe effettuare un salto alla istruzione indicata dalla label «START». Il codice macchina quindi deve utilizzare l'indirizzo reale che corrisponde alla label START (in questo caso 07F0).

L'ultima istruzione, JMP LOOP, ha un codice del tutto simile. L'unica differenza è che questo salto non dipende da alcuna condizione. Il codice per tale tipo di salto è C3; l'indirizzo (07F2) è memorizzato come nel caso dell'istruzione JZ.

Notate che il programma in linguaggio macchina è costituito da una serie di byte, ciascuno dei quali può avere un solo significato tra i tre possibili: codice operativo, dato ed indirizzo. Per sapere con quale tipo di byte si ha a che fare, è necessario che conosciate il contesto in cui è inserita l'informazione. Il microprocessore, poiché sa se un certo codice operativo deve essere eseguito da un dato o da un indirizzo, è in grado di sapere sempre che tipo d'informazione sta trattando.

CONFRONTO TRA I DIVERSI TIPI DI LINGUAGGI

I linguaggi ad alto livello sono quelli che il programmatore può utilizzare con maggior facilità e sono dei linguaggi che possono essere indipendenti dal particolare tipo di microprocessore. Per tradurre i programmi in codice macchina è necessario che nella memoria del microcalcolatore siano immagazzinati dei programmi di traduzione molto lunghi. I linguaggi ad alto livello sono inoltre meno efficienti per quanto riguarda la velocità di esecuzione e l'occupazione della memoria. Un programma equivalente scritto in linguaggio di assemblaggio viene infatti eseguito più velocemente ed occupa meno memoria.

Il linguaggio di assemblaggio è oggi largamente usato per programmare i microcalcolatori. È più difficile scrivere dei programmi in linguaggio di assemblaggio di quanto non lo sia scriverli in un linguaggio ad alto livello; l'operazione di tradurre in linguaggio macchina è tuttavia molto più facile se si parte dal linguaggio di assemblaggio piuttosto che da un linguaggio ad alto livello. In quelle applicazioni in cui il programma debba essere eseguito alla massima velocità o, alternativamente, debba occupare il minor spazio di memoria, la scelta del linguaggio di assemblaggio è di solito la migliore possibile. Da un punto di vista educativo infine la programmazione in linguaggio d'assemblaggio vi avvicina molto di più al modo di operare di un sistema a microprocessore.

Il linguaggio macchina è il solo linguaggio che il microprocessore può direttamente interpretare. È però scomodo per le persone che devono utilizzarlo ed è difficile programmare direttamente in tale linguaggio. Di solito i programmi vengono scritti in linguaggio di assemblaggio e solo successivamente vengono tradotti in linguaggio macchina. La traduzione è effettuata da un programma speciale che è detto *assembler* (assemblatore).

LA PROGRAMMAZIONE DEL μLAB

Per programmare il μLab, bisogna assemblare «a mano» i programmi scritti in linguaggio d'assemblaggio, facendo riferimento alla tabella delle istruzioni (Appendice B) che riporta il codice macchina (in esadecimale) per le diverse istruzioni. Nel μLab possono essere introdotti solo dei programmi scritti in linguaggio macchina. In generale perciò voi scriverete dei programmi in linguaggio di assemblaggio e li tradurrete poi in codice macchina (in esadecimale). Il codice macchina può così essere introdotto nel μLab. Per il codice macchina viene usata la rappresentazione esadecimale poiché è più facile da utilizzare di quanto non lo sia quella binaria. Il μLab contiene poi dei programmi che effettuano la traduzione, abbastanza semplice, da esadecimale a binario.

Interpretazione del contenuto della memoria

INTRODUZIONE

In questo esperimento esaminerete il contenuto di una parte della memoria del Lab e tradurrete quindi il codice macchina in linguaggio di assemblaggio (che è l'operazione opposta a quella solita di assemblaggio).

PROCEDIMENTO

A) Accendete il μLab, se già non lo fosse.

B) Il display dovrebbe visualizzare uLAb UP . Se è così, premete [RESET].

C) Premete [FETCH ADRS] [0] [7] [F] [0] . Con questa operazione indicate al μLab che volete esaminare l'indirizzo 07F0. Il dato contenuto a questa locazione (3E) è riportato nella colonna *Contenuto* della Tabella 3-5.

D) Premete [STORE /INCR] in modo da esaminare la locazione di memoria successiva.

E) Trascrivete il dato visualizzato sul display nella colonna *Contenuto* della Tabella 3-5.

F) Ripetete i passi D ed E finché non avete riempito tutti gli spazi disponibili nella colonna *Contenuto* della Tabella 3-5.

G) Andate a vedere la Tabella B-1 nell'Appendice B, in cui sono riportati tutti i codici operativi e i loro codici mnemonici corrispondenti. Notate che vi sono due elenchi: uno suddiviso per funzioni, l'altro in cui i codici operativi sono elencati in ordine numerico (sul retro della pagina).

Indirizzo	Contenuto	Mnemonico / Dato / Indirizzo	
07F0	3E	codice operativo	
07F1		dato	
07F2		codice operativo	
07F3		codice operativo	
07F4		dato	
07F5		codice operativo	
07F6		indirizzo	
07F7		indirizzo	
07F8		codice operativo	
07F9		indirizzo	
07FA		indirizzo	

Tabella 3-5. Elenco delle istruzioni per l'Esperimento 3-2

ESPERIMENTO 3-2

(continuazione)

H) Facendo uso della tabella che vi dà l'elenco dei codici operativi in ordine numerico, riportate i codici mnemonici nella Tabella 3-5. Ricordatevi che non tutte le locazioni di memoria contengono un codice operativo. Parecchie istruzioni sono costituite da un codice operativo seguito da un dato o da un indirizzo di salto.
Perché vi sia più facile decodificare i codici macchina, nella Tabella 3-5 sono indicati quali byte sono dei codici operativi, quali sono dei dati e quali sono degli indirizzi. Voi dovete decodificare solo i codici operativi; copiate invece pari pari nella colonna di destra i dati e gli indirizzi di salto.

I) Ritornate a guardare il listing del programma riportato nella Tabella 3-4. Tale listing in linguaggio d'assemblaggio dovrebbe coincidere con quello che voi avete appena ricavato e trascritto nella Tabella 3-5.

RIASSUNTO

Partendo dalle informazioni memorizzate nella ROM del μLab è stato ricavato un listing. Avete così tradotto i codici in mnemonici. In effetti questa è un'operazione di traduzione da linguaggio macchina a linguaggio di assemblaggio ed è detta *assemblaggio inverso* o *disassemblaggio* (Disassembly). Le informazioni contenute nella memoria del microprocessore possono essere dei codici operativi, dei dati o degli indirizzi di salto a seconda del contesto in cui si trovano.

UN ESEMPIO DI APPLICAZIONE

Analizziamo ora una tipica applicazione che utilizzi un microprocessore ed esaminiamo il programma necessario. Supponete di avere un nastro trasportatore che, all'interno di una fabbrica, trasferisca degli ingranaggi dal reparto produzione al reparto imballaggio. All'estremità del nastro trasportatore gli ingranaggi vanno a cadere nel cartone di imballaggio. Un operatore deve contare gli ingranaggi man mano che cadono nella scatola e, quando in un cartone si sono accumulati dieci ingranaggi, deve sostituire tale cartone pieno con uno vuoto. Vediamo come è possibile automatizzare questo sistema.

Per prima cosa, sarà necessario dell'hardware (Figura 3-6). Si può utilizzare una fotocellula (il sensore degli ingranaggi) per generare un impulso ad ogni ingranaggio che passa sul nastro, in modo tale che essi possano essere contati. Serve anche un altro nastro trasportatore per le scatole, cosicché una scatola piena possa essere automaticamente fatta avanzare e sostituita da una scatola nuova. Da ultimo servirà un elemento di controllo (controllore) che conti gli ingranaggi e sposti le scatole quando necessario.

Per contare i dieci impulsi e per inviare quindi un impulso al nastro che trasporta le scatole si potrebbe costruire un controllore fatto ad hoc, utilizzando della logica digitale standard. È questo l'approccio della logica sparsa. Alternativamente, per svolgere tale compito, si può utilizzare un microcalcolatore, nel qual caso servirà un programma che indichi al microcalcolatore quello che deve essere fatto.

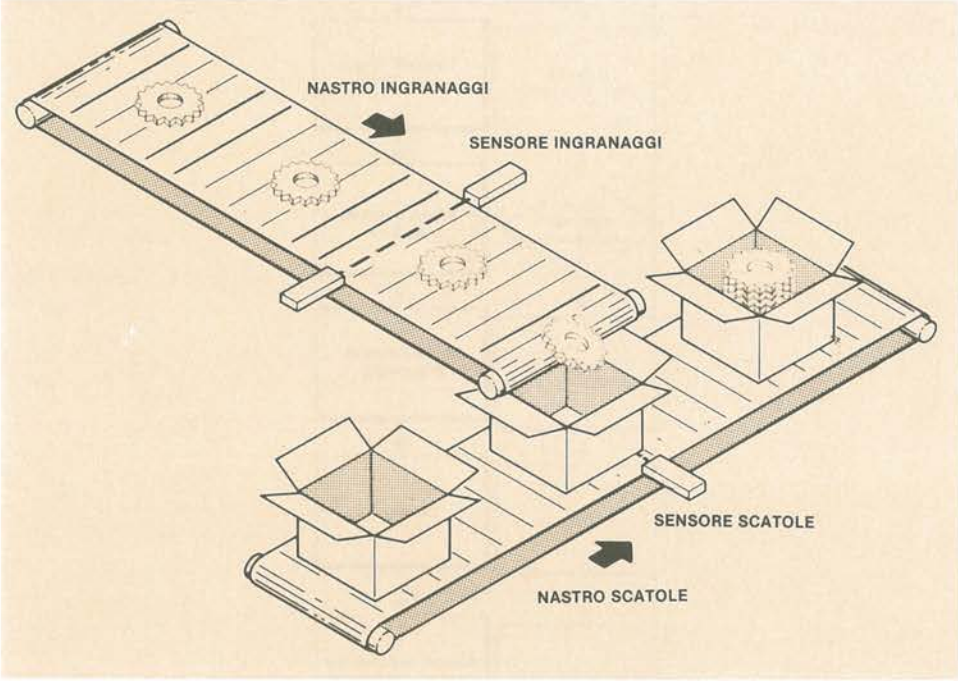


Figura 3-6. Sistema con nastro trasportatore

La Figura 3-7 riporta il diagramma di flusso di tale programma. Per prima cosa viene fermato il nastro trasportatore degli ingranaggi e viene messa in posizione una nuova scatola. Viene poi fatto ripartire il nastro trasportatore degli ingranaggi che vengono man mano contati. Il processo viene ripetuto quando sono passati dieci ingranaggi.

Nel diagramma di flusso la parola *Count* (Conteggio o meglio contatore) non si riferisce ad un contatore vero e proprio, ma è solo il nome che abbiamo assegnato ad una locazione di memoria. Così "Metti a zero il contatore" vuol dire che in questa locazione di memoria viene caricato il valore zero. Ancora, "Incrementa Count" significa che viene sommato uno al valore contenuto in quella locazione di memoria. Questi tipi di operazioni sono del tutto generali (non sono cioè operazioni specifiche di questa applicazione) e sono compatibili con le istruzioni che i microprocessori possono eseguire.

Questa applicazione vi mostra come sia possibile sostituire dell'hardware facendo uso di programmi (software). Il sistema non contiene nessun particolare dispositivo hardware che operi come contatore. Piuttosto viene utilizzata una locazione non specializzata della memoria e ci si avvale delle capacità aritmetiche del microprocessore. Questo tipo di funzione, nei sistemi a microprocessore, è comunemente realizzata attraverso il software, mentre nei sistemi a logica sparsa viene a richiedere dei circuiti hardware.

MODIFICARE UN PROGRAMMA

Nel diagramma di flusso della Figura 3-7 è presente un problema. Che cosa si verifica se non c'è alcuna scatola in posizione? Il nastro trasportatore finirebbe per scaricare ingranaggi sul pavimento. L'operaio addetto al controllo si accorgerebbe che non funziona qualcosa: il microcalcolatore invece deve essere esplicitamente programmato affinché possa rilevare tale situazione ed effettuare qualche operazione di correzione.

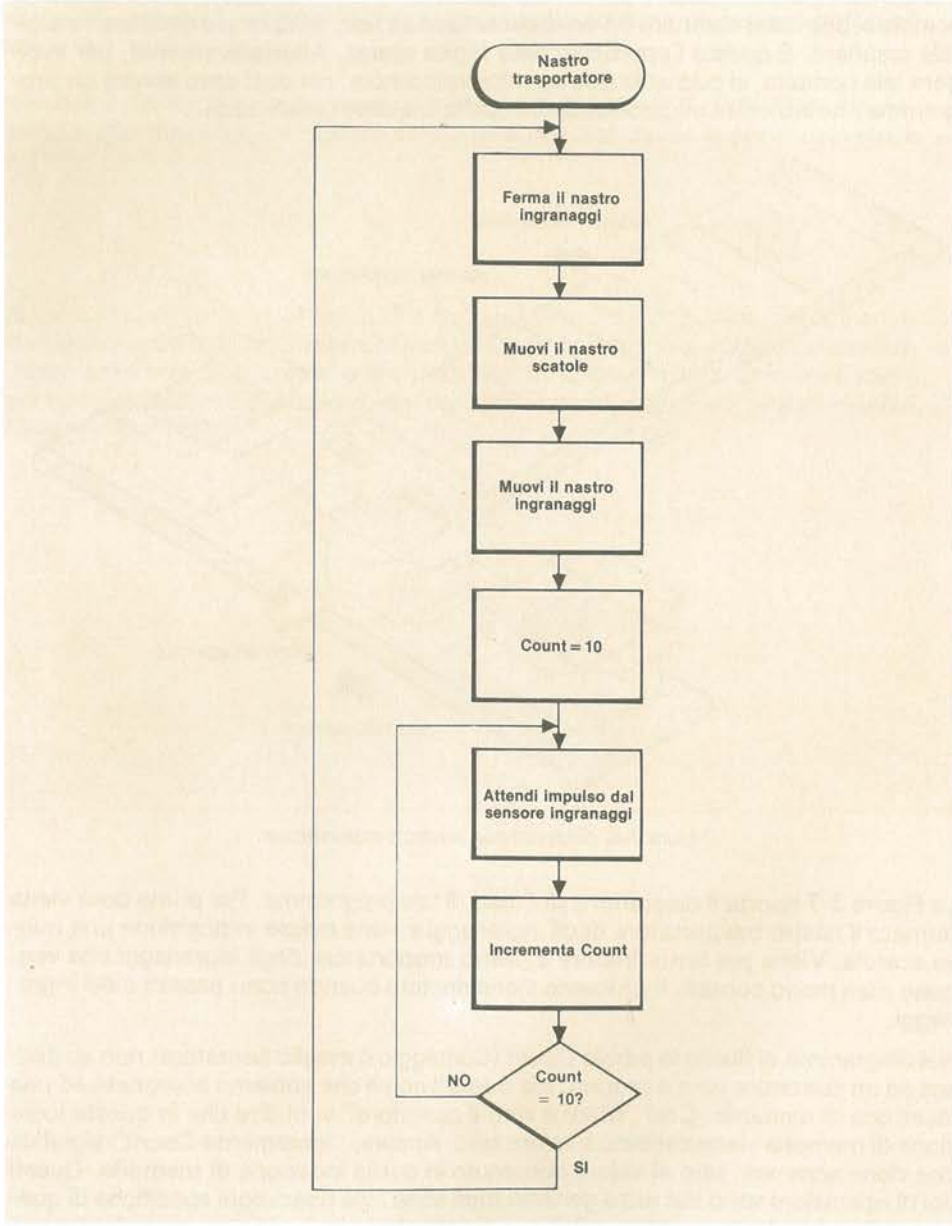


Figura 3-7. Diagramma di flusso del programma di controllo nastro trasportatore

La Figura 3-8 riporta un diagramma di flusso modificato in grado di svolgere anche questa funzione. Dopo che il microprocessore ha segnalato al nastro trasportatore delle scatole di muoversi, il nastro degli ingranaggi viene fermato finché una scatola vuota non è arrivata in posizione (rilevata da una seconda fotocellula).

Utilizzando il diagramma di flusso è più facile notare e risolvere un problema di questo tipo, di quanto non sarebbe se si lavorasse direttamente con il programma vero e proprio. Quando credete che il diagramma di flusso sia corretto, potete incominciare a scrivere il programma.

Notate che se aveste costruito il controllore in logica cablata, si sarebbe dovuto ricorrere a modifiche hardware, sicuramente più difficili da realizzare.

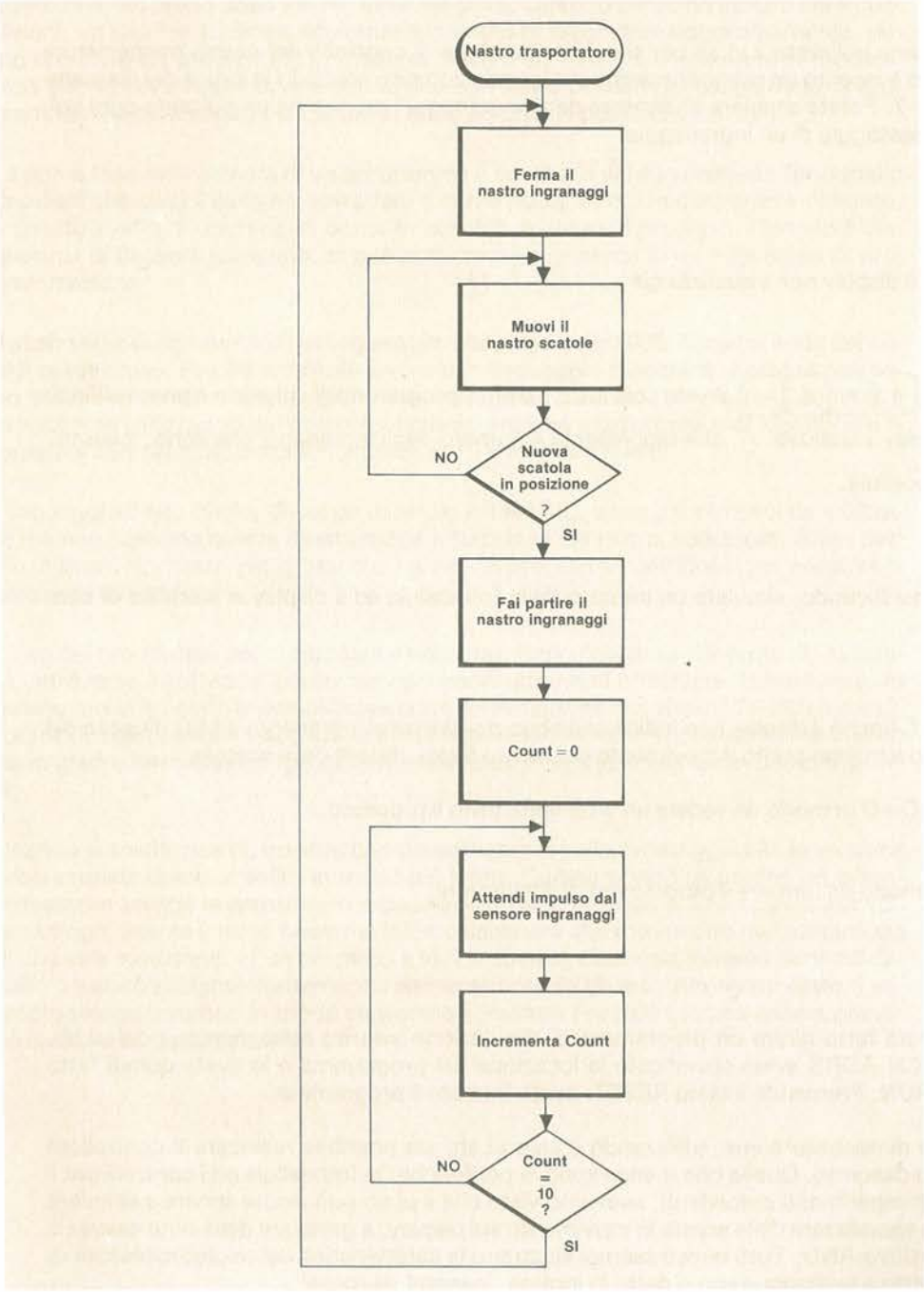


Figura 3-8. Diagramma di flusso del controllore modificato nastro trasportatore

ESPERIMENTO 3-3

Simulazione del nastro trasportatore

INTRODUZIONE

In questo esperimento, viene utilizzato il μLab per simulare l'unità per il controllo del nastro trasportatore appena descritta. Nel μLab è inserito un programma che svolge tale compito secondo la logica del diagramma di flusso della Figura 3-7. Potete simulare il "sensore degli ingranaggi" premendo un pulsante ogni volta che volete segnalare il passaggio di un ingranaggio.

PROCEDIMENTO

A) Premete [RESET], se il display non visualizza già uLAb UP.

B) Premete [FETCH ADRS] [0] [4] [F] [8] [RUN]. Avete così fatto partire il programma il cui inizio è posto all'indirizzo 04F8. Il display visualizza 0 che rappresenta il numero degli ingranaggi che sono "passati" attraverso la fotocellula.

C) Premete [0]. Così facendo, simulate un impulso della fotocellula ed il display si modifica di conseguenza.

D) Ripetete il passo C finché il display non indica che sono passati dieci ingranaggi. I LED d'uscita del μLab simuleranno a questo punto il movimento del nastro trasportatore delle scatole.

E) Ripetete il passo C e D in modo da vedere un'altra volta tutto il processo.

F) Premete [RESET], in modo da fermare il programma di simulazione.

RIASSUNTO

In questo esperimento avete fatto girare un programma di simulazione inserito nella memoria del μLab. Servendovi del tasto FETCH ADRS avete specificato la locazione del programma e lo avete quindi fatto partire premendo il tasto RUN. Premendo il tasto RESET, avete fermato il programma.

Questo esperimento vi ha dimostrato come, utilizzando il solo μLab, sia possibile realizzare il controllore specializzato che avevamo descritto. Quello che manca sono le periferiche: la fotocellula ed i controlli per il nastro trasportatore. Negli esperimenti precedenti, avevamo visto che il μLab può anche servire a simulare la partenza di un missile, a visualizzare delle scritte in movimento sul display, a generare delle note casuali o ad operare come un dispositivo AND. Tutti questi esempi illustrano la caratteristica del microprocessore di essere un componente adatto a qualunque uso o detto in inglese "general-purpose".

Un programma è un elenco dettagliato di istruzioni che dice al microprocessore quello che, un passo dopo l'altro, deve eseguire. Quando vengono variate delle condizioni, un sistema a microcalcolatore è in grado di rispondere opportunamente, posto che ci sia un programma in grado di rilevare tali eventi e di generare le risposte. Non può tuttavia rispondere a eventi che non siano previsti dal programma. Il programmatore deve perciò conoscere in anticipo tutte le possibili situazioni.

La prima fase nella stesura di un programma è costituita dal suo progetto funzionale: decidere che cosa il sistema dovrà fare e come potrà farlo. Un diagramma di flusso, a questo stadio, è un metodo comodo per documentare il progetto. Quando il diagramma di flusso è completo, si può scrivere il programma in un linguaggio di programmazione.

Il μLab viene programmato nel linguaggio macchina dello 8085, introducendo dei codici esadecimali. Poiché è difficile lavorare in linguaggio macchina, i programmi sono dapprima scritti in linguaggio di assemblaggio. Questo vi permette di scrivere il programma utilizzando dei codici mnemonici anziché esadecimali e di identificare le locazioni con dei nomi simbolici anziché con i loro indirizzi reali.

I linguaggi ad alto livello, di cui un esempio è il BASIC, sono più semplici da utilizzare ma non possono essere direttamente interpretati dal microprocessore. Sono perciò utilizzati su sistemi più grossi che hanno dei programmi particolari per eseguire le istruzioni ad alto livello.

L'uso dei programmi per controllare il sistema microcalcolatore permette di realizzare, attraverso il software, parecchie operazioni altrimenti effettuate da hardware. In questo modo è possibile semplificare considerevolmente l'hardware del sistema e disporre di una flessibilità eccezionale. Gli esperimenti hanno mostrato come il μLab sia in grado, introducendo programmi differenti, di svolgere numerosi compiti diversi.

Quando si sostituisce l'hardware con del software si ha lo svantaggio che la versione programmata opera di solito in modo più lento. Questo si verifica perché un microprocessore esegue le istruzioni in modo sequenziale, facendo solo un passo alla volta. Ad ogni istante è tutto il sistema microcalcolatore che è coinvolto nell'esecuzione di una sola istruzione. In un progetto a logica sparsa, viceversa, diverse parti del circuito possono svolgere contemporaneamente compiti diversi. Altrimenti detto, i microprocessori lavorano in modo sequenziale, mentre i circuiti a logica sparsa possono lavorare in modo parallelo.

DOMANDE

Lezione 3

1. Il linguaggio che il microprocessore può interpretare direttamente è detto:

   a. linguaggio di assemblaggio.
   b. linguaggio ad alto livello.
   c. linguaggio macchina.
   d. linguaggio BASIC.

2. Quando scrivete un programma per il μLab, dovrete prima scriverlo in linguaggio

   ——— e poi tradurlo in linguaggio ———.

3. Rispetto ad un dispositivo AND convenzionale, il dispositivo AND realizzato con microprocessore ha il vantaggio di:

   a. essere più veloce.
   b. essere più semplice da modificare.
   c. semplificare l'hardware.
   d. assorbire meno corrente.

4. In un programma scritto in linguaggio di assemblaggio, lo scopo delle label è quello di:

   a. dare un nome al programma.
   b. spiegare la funzione delle istruzioni.
   c. indicare il punto di partenza del programma.
   d. identificare delle locazioni cui altre istruzioni fanno riferimento.

5. Il μLab può realizzare diverse funzioni in quanto:

   a. il suo modo di operare è controllato da software.
   b. è un sistema digitale.
   c. fa uso di una RAM.
   d. è controllato da tastiera.

II
INTRODUZIONE ALLA PROGRAMMAZIONE

In questa sezione verranno presentati alcuni programmi e verranno introdotti concetti di software. Saranno introdotti alcuni esperimenti in modo da darvi familiarità con il μLAB.

Il μLAB è un piccolo microcalcolatore progettato appositamente per fini didattici, che contiene un microprocessore 8085, 2K byte di ROM e 1K byte di RAM. Nel sistema sono compresi una tastiera, attraverso la quale si possono inserire programmi, memorizzare dati e dare comandi per controllare le operazioni del microcalcolatore, e un display che vi permette di vedere i contenuti della memoria e dei registri. Sono presenti ancora una porta d'uscita con un LED per ognuna delle sue otto linee e una porta d'ingresso, su ciascuna linea della quale è posto un interruttore a slitta per un totale di otto interruttori. Sulla scheda è presente anche un altoparlante controllato direttamente dal microprocessore. Degli altri LED infine sono collegati al bus degli indirizzi, al bus dei dati e alle principali linee di controllo al fine di mostrare la loro attività.

La ROM presente nel μLAB contiene i programmi per leggere la tastiera, quelli per eseguire i comandi della tastiera stessa e quelli per visualizzare dati sul display. Tutte le operazioni del sistema sono controllate da questo software, detto *monitor*. Gli esperimenti che seguono servono a mostrare la possibilità del monitor e l'hardware da esso controllato.

LEZIONE 4

Uso del Microprocessor lab

INTRODUZIONE

Le operazioni principali che possono essere eseguite utilizzando il Microprocessor Lab sono: la scrittura dei dati nella memoria, l'esame dei contenuti della memoria e l'esecuzione dei programmi. I programmi possono essere eseguiti alla velocità normale, come negli esperimenti precedenti, oppure un passo per volta in modo da permettervi di seguire dettagliatamente le operazioni che vengono effettuate. In questa lezione verranno illustrate queste possibilità.

LA MAPPA DI MEMORIA DEL μLAB

La *mappa di mèmoria* del μLab (Fig. 4-1) mostra quali indirizzi siano assegnati ai diversi dispositivi. La ROM occupa gli indirizzi 0000-07FF mentre la RAM va da 0800 a 0BFF. Si noti, però, che non tutta la RAM è utilizzabile dai vostri programmi.

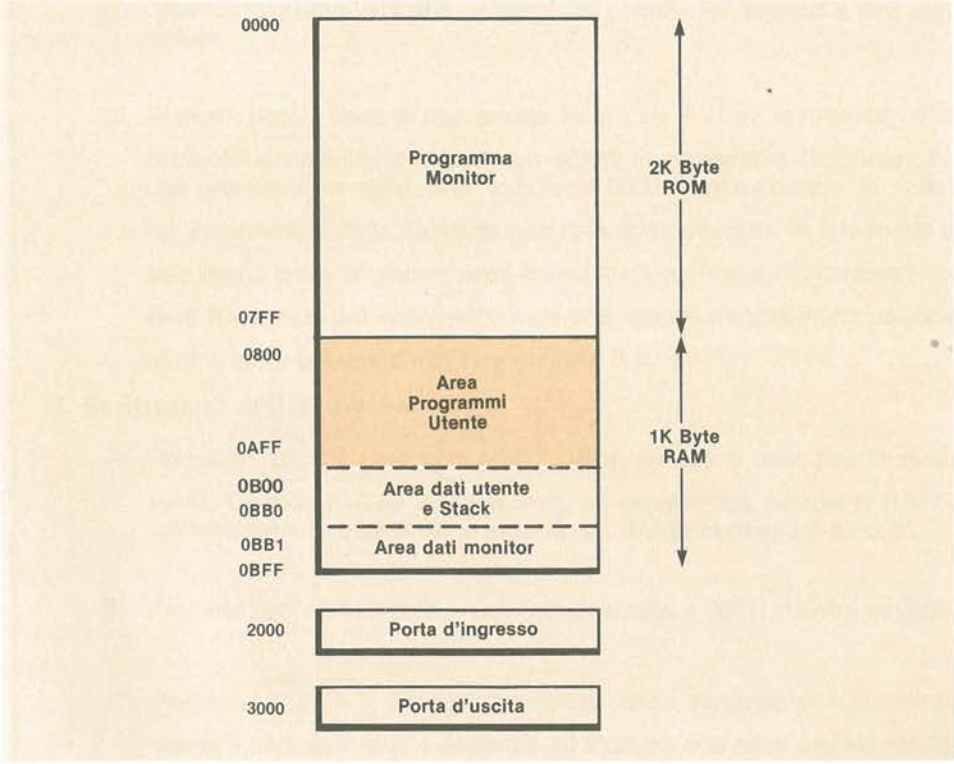


Figura 4-1. Mappa di memoria del Microprocessor Lab

La parte alta della RAM da 0BB1 a 0BFF è utilizzata dal programma monitor e non deve perciò essere utilizzata da altri programmi. Le locazioni 0AEF-0BB0 sono invece riservate ad altri scopi di cui verrà data descrizione in seguito. L'area da 0800 a 0AEE è invece disponibile per i vostri programmi.

La porta d'ingresso è posta all'indirizzo 2000 mentre quella d'uscita è posta all'indirizzo 3000. Altre porte presenti, utilizzate per controllare la tastiera, il display e il circuito di «single step» (passo singolo) saranno descritte nella sezione III, Hardware del Sistema a Microprocessore.

ESPERIMENTO 4-1

Esame della memoria e memorizzazione dei dati

INTRODUZIONE

In questo esperimento, per esaminare e modificare il contenuto della memoria del μLab, vengono utilizzati tre tasti. Il tasto FETCH ADRS vi serve per poter esaminare dati immagazzinati in un indirizzo qualunque. Con il tasto STORE/INCR potrete memorizzare dati o esaminare successivamente locazioni di memoria. Entrambi questi tasti sono già stati utilizzati nella Lezione 3, ma questo esperimento ne esaminerà più in dettaglio il modo di operare. Sempre in questo esperimento viene introdotto il tasto DECR (decrementa) che vi permette di «andare all'indietro» nella memoria.

PROCEDIMENTO

I. Esame del contenuto della memoria

A) Sul display dovrebbe comparire uLAb UP ad indicare che il sistema sta nel suo modo normale di «attesa comando». Se così non fosse, premete [RESET].

B) Premete [FETCH ADRS] [0] [0] [0] [0]. Nelle quattro cifre di sinistra del display è ora visualizzato l'indirizzo (0000) che avete appena introdotto, mentre nelle due cifre di destra è presentato il dato (26) contenuto a tale indirizzo. Si ricordi che indirizzi e dati sono mostrati in forma esadecimale.

C) Premete [STORE /INCR]. Questa operazione fa sì che il μLab memorizzi all'indirizzo visualizzato il dato presente sul display e che venga quindi incrementato l'indirizzo. Poiché non avete fatto altro che prelevare un dato dalla locazione 0000 e memorizzato lo stesso dato alla stessa locazione, il contenuto della memoria non sarà stato alterato. In tale modo di operare il tasto [STORE /INCR] serve solo come tasto di «incremento indirizzo» e permette di esaminare locazioni di memoria successiva. Ripetendo più volte [STORE /INCR] potete così esaminare una intera sezione di memoria: questa possibilità è stata già usata nell'Esperimento 3-2.

II. Scrittura di dati in memoria

A) Premete [FETCH ADRS] [0] [8] [0] [0]. Con questo si specifica l'indirizzo 0800 che è l'inizio della RAM. Quando il μLab viene acceso, un programma riempie la RAM di tutti zero, cosicché questa locazione (e così tutte le altre fino a 0AEE) contiene il dato 00.

B) Premete [STORE /INCR]. L'indirizzo è così incrementato a 0801, mentre in 0800 è stato lasciato 00.

C) Premete [C] [3]. Si noti che questi valori vengono visualizzati nella parte dati del display e che si è acceso il punto decimale ad indicare che siete ora nel modo «inserimento dati».

ESPERIMENTO 4-1

(continuazione)

D) Premete [STORE /INCR]. Nella locazione di indirizzo 0801 viene così immagazzinato il dato C3, mentre l'indirizzo viene incrementato a 0802. Si noti che il punto decimale si è spento a indicare che il dato inserito da tastiera è stato memorizzato e che il μLab non è più nel modo «inserimento dati».

E) Premete [STORE /INCR]. Con questa operazione si lascia 00 in 0802 e si passa all'indirizzo 0803.

F) Premete [8] [STORE /INCR]. Si noti che non è necessario premere 0 prima di 8: se si inserisce una sola cifra il μLab inserisce automaticamente uno zero prima di questa. Avete così memorizzato questi dati:

Indirizzo	Contenuto
0800	00
0801	C3
0802	00
0803	08

G) Premete [DECR]. Così facendo decrementate l'indirizzo visualizzato sul display e potete controllare i dati che avete memorizzato.

H) Ripetete il passo G finché non avete verificato che tutti i valori indicati sono correttamente contenuti in memoria. Si noti che [DECR] non modifica mai il contenuto della memoria. Si poteva effettuare tale verifica anche premendo [FETCH ADRS] [0] [8] [0] [0] e utilizzando il tasto [STORE /INCR] per esaminare le locazioni successive.

III Correzione di errori

A) Premete [FETCH ADRS] [0] [0] [0] [0]. Questo valore corrisponde all'indirizzo della prima locazione della ROM.

B) Premete [0] [STORE /INCR]. Questa operazione cerca di memorizzare 00 in questa locazione.

C) Che cosa è successo? Avete tentato di scrivere nella ROM, cosa impossibile. Il monitor rileva tale errore, emette un suono di avvertimento, non incrementa l'indirizzo e vi mostra il dato contenuto in questa locazione.

D) Questo è un errore che il monitor ha potuto rilevare. Ma che cosa si può fare quando si verificano errori del tipo inserire dati o indirizzi sbagliati? Quando inserite un indirizzo sbagliato, potete sempre premere [FETCH ADRS] una seconda volta e ripartire da capo.
Ad esempio premete [FETCH ADRS] [0] [7]. A questo punto vi accorgete che vi interessa l'indirizzo 0900. Anche se siete a metà dell'operazione di inserimento, potete ricominciare a scegliere il nuovo indirizzo premendo [FETCH ADRS] [0] [9] [0] [0].

E) Ora che è selezionato l'indirizzo corretto potete inserire un dato. Premete [7] [9]. Questo valore comparirà nelle due cifre di destra del display.

F) Supponiamo che non sia questo il dato che voi volete. Per esempio volete introdurre 69. È sufficiente che, di seguito, inseriate il nuovo dato: il vecchio viene fatto scorrere via. Durante l'operazione di inserimento dati potete introdurre un numero a piacere di cifre, il μLab terrà conto solo delle ultime due. Il punto decimale è acceso a indicare che il sistema è ancora in attesa di dati. Quando premete [STORE /INCR], però, il dato viene immagazzinato in memoria e il punto decimale si spegne.

G) Premete [STORE /INCR]. Avrete così trasferito 69 nell'indirizzo 0900. Supponiamo ora che vogliate modificarlo in 68.

H) Premete [DECR]. Sul display compaiono l'indirizzo 0900 e il dato 69. Premete i tasti corrispondenti al dato corretto, 68, e poi chiudete l'operazione con [STORE /INCR]. Avrete così cambiato il dato.

RIASSUNTO

Nello svolgimento di questo esperimento avete letto parte della ROM ed avete memorizzato dei dati nella RAM. Avete usato i tasti:

[FETCH ADRS] vi permette di definire un certo indirizzo e vedere il dato in esso memorizzato.

[STORE /INCR] memorizza il dato presente sul display all'indirizzo visualizzato e mostra quindi la locazione successiva.

[DECR] decrementa l'indirizzo presente sul display, senza modificare il dato e visualizza la locazione precedente.

È con questi tre tasti che vi diventa facile immagazzinare ed esaminare dati nella memoria del μLab.

UN PROGRAMMA SEMPLICE

In effetti il gruppo di dati, che avete memorizzato in RAM nel precedente esperimento, costituisce un programma, il cui listing è riportato nella tabella.

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	00	START:	NOP	;Nessuna operazione
0801	C3		JMP START	;Salta all'inizio
0802	00			
0803	08			

Tabella 4-1. Listing del Programma

Al codice 00 (presente all'indirizzo 0800) corrisponde il simbolo mnemonico NOP, ovvero Nessuna Operazione (in inglese: No Operation, spesso abbreviato in no-op). Come dice il nome, tale istruzione non fa nulla. E allora a che cosa serve? In realtà è utile per due motivi. In primo luogo, dato che è necessario un certo tempo per prelevare e decidere di che istruzione si tratta, può essere utilizzata per realizzare un breve intervallo di ritardo quando si vuole che non venga eseguita alcuna operazione. In effetti però serve soprattutto come "tappabuchi". Se voi, dopo aver scritto un programma, volete riprenderlo dall'inizio e aggiungere istruzioni, vi è sufficiente sostituire ai NOP presenti nel programma le vostre nuove istruzioni. Se non avete messo alcun NOP nel vostro programma, anche per inserire una sola istruzione potreste essere costretti a spostare blocchi interi di programma.

L'altra istruzione presente nel programma è un'istruzione di salto (jump). Il codice C3 (mnemonico JMP) significa che deve essere effettuato un salto all'indirizzo specificato dai due byte di memoria successivi. Il *byte meno significativo di tale indirizzo* di salto è memorizzato subito dopo il codice operativo; poi il byte più significativo. Ad esempio, dell'indirizzo 0800, 00 è memorizzato in 0802 e 08 in 0803.

Ovviamente questo programma non fa proprio nulla se non continuare a girare su se stesso all'infinito. Il suo diagramma di flusso è riportato in Fig. 4-2.

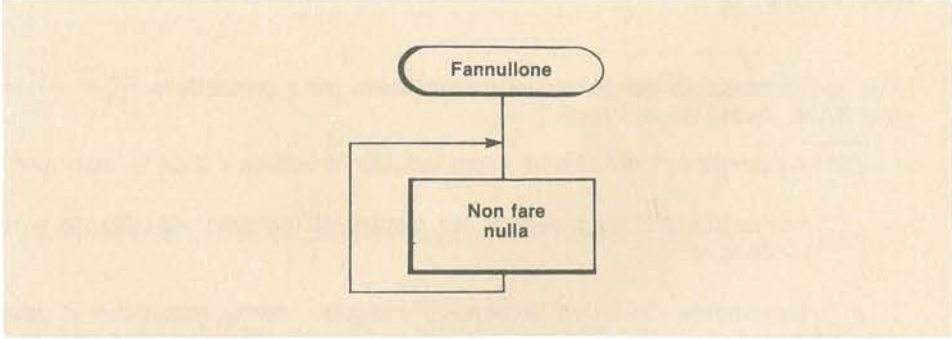


Figura 4-2. Diagramma di flusso del programma "Fannullone"

ESPERIMENTO 4-2

Esecuzione dei programmi

INTRODUZIONE

Con il μLab è possibile eseguire un programma in tre modi: utilizzando la solita funzione RUN, che fa girare il programma alla sua velocità normale, e in due diversi modi di single-step che vi permettono di eseguire il programma una istruzione (o parte di una istruzione) alla volta, ovvero passo passo. Il tasto RESET è utilizzato per arrestare l'esecuzione di un programma e per uscire dal modo hardware step.

PROCEDIMENTO

I. Esecuzione del programma utilizzando l'Instruction Step

A) Premete [FETCH ADRS] [0] [8] [0] [0] . Si verifichi che in questa locazione sia contenuto lo stesso dato della prima linea della Tabella 4-1.

B) Premete [STORE /INCR]. Si verifichi il dato presente in questa locazione.

C) Si ripeta il passo B finché non si sia verificato che il programma della Tabella 4-1 è tutto memorizzato correttamente. Se non lo fosse si ripeta la parte II dell'Esperimento 4-1.

D) Premete [FETCH ADRS] [0] [8] [0] [0] in modo da riportare l'indirizzo all'inizio del programma (0800). Alternativamente potete servirvi del tasto [DECR] finché non sia visualizzato il valore 0800.

E) Premete [INSTR STEP]. Con questa operazione si provoca l'esecuzione dell'istruzione presentata sul display e la visualizzazione dell'istruzione successiva. Anche se il modo di operare di questo tasto può sembrare molto simile a quello di [STORE /INCR], la funzione svolta è in realtà del tutto diversa. Quando si utilizza [STORE /INCR] vengono esaminati solo dei contenuti di memoria. Premendo il tasto [INSTR STEP] invece, il contenuto della locazione di memoria visualizzata viene interpretato come una istruzione e come tale eseguito dal microprocessore. Quando ci si serve del μLab per eseguire un'istruzione (o un programma) attraverso i tasti [INSTR STEP], [HDWR STEP] o [RUN], il display deve visualizzare una locazione che contenga un codice operativo. L'esecuzione deve iniziare da una locazione che non contenga un dato o un indirizzo. Per esempio l'esecuzione del programma della Tabella 4-1 può iniziare dalle locazioni 0800 o 0801 ma non da 0802 o 0803.

F) Sul display è ora presentata l'istruzione di salto, C3. Premete [INSTR STEP]. Poiché si tratta di una istruzione di salto l'indirizzo salta appunto al valore 0800. Si noti che le locazioni di memoria che contengono l'indirizzo del salto non sono mai visualizzate. Questo accade poiché quando voi premete [INSTR STEP], il μLab esegue tutta l'istruzione e di essa fanno parte anche i due byte dell'indirizzo del salto.

G) Premete [INSTR STEP] più volte per vedere il loop del programma.

ESPERIMENTO 4-2

(continuazione)

II. Esecuzione del programma utilizzando l'Hardware Step

A) Premete [FETCH ADRS] [0] [8] [0] [0]. Con questa operazione si porta l'indirizzo all'inizio del programma.

B) Premete [HDWR STEP]. Poiché il tasto di hardware step ferma completamente il processore al termine dell'esecuzione, il display si spegne. [INSTR STEP], invece, fa sì che venga eseguita un'istruzione riportando poi il controllo al programma monitor.

C) Si osservino i 16 LED indicati con ADDRESS. Questi LED sono direttamente collegati al bus degli indirizzi e in questo momento mostrano 0000 1000 0000 0000. Servendovi della tabella binario-esadecimale posta sul circuitc stampato del μLab, a destra dei LED e degli indirizzi, convertite questo valore in esadecimale e vedrete così che coincide con l'indirizzo da voi definito nel passo A.

D) Osservate gli 8 LED indicati con DATA. Questi LED sono collegati al bus dei dati e mostrano il dato 00 memorizzato all'indirizzo 0800.

E) Osservate i 6 LED indicati con STATUS. Questi LED indicano se è stata effettuata una operazione di lettura o scrittura e se tale operazione ha coinvolto la ROM, la RAM, la porta di ingresso o la porta d'uscita. Sono accesi i LED RAM e READ (lettura) ad indicare appunto che il dato è stato letto dalla RAM.

F) Premete [HDWR STEP]. L'indirizzo si incrementa e i LED dei dati e di stato mostrano le informazioni corrispondenti al nuovo indirizzo. Viene eseguita l'istruzione, esattamente come nel caso di [INSTR STEP].

G) I LED dei dati mostrano ora il codice operativo del salto, C3 (1100 0011). Premete [HDWR STEP]. Si noti che viene incrementato l'indirizzo ma non viene eseguito il salto. Si vede così una differenza essenziale tra i due modi di esecuzione: con [HDWR STEP] si procede in una locazione di memoria per volta, mentre con [INSTR STEP] si procede di una istruzione per volta (anche se l'istruzione è costituita da più di un byte). Con [HDWR STEP] l'istruzione non è eseguita finché non sono state lette tutte le parti che compongono l'istruzione (p.es. una istruzione a tre byte non viene eseguita finché non si è giunti al terzo passo).

H) Premete [HDWR STEP] [HDWR STEP]. Con questo viene completata l'istruzione di salto. I LED degli indirizzi mostrano ora 0800, ad indicare che è stata eseguita l'istruzione di salto.

I) Premete [HDWR STEP] ripetutamente e osservate che il programma continua a ripetersi.

J) Premete [RESET]. Così facendo si ripristina il modo di operare solito del sistema. Quando si preme [RESET], viene eseguita l'istruzione che stava aspettando di essere eseguita (mostrata dai LED dei dati) e il display visualizzerà quindi l'istruzione successiva.

III. Esecuzione del Programma (Run)

A) Premete [FETCH ADRS] [0] [8] [0] [0] per impostare l'indirizzo di partenza.

B) Premete [RUN]. Il programma sta ora girando alla velocità normale (circa 2 microsecondi per istruzione).

C) Osservate i LED degli indirizzi. Sembrano accesi in modo da indicare 0803 (0000 1000 0000 0011). In realtà gli indirizzi passano da 0800 a 0803 per poi ricominciare da 0800. È la stessa sequenza di programma che avete visto passo passo finora. Adesso però i LED si modificano così velocemente che sembrano sempre accesi.

D) Osservate i LED di stato. Indicano una lettura in RAM, esattamente come nel modo hardware step. Poiché le istruzioni accendono gli stessi LED di stato (il programma legge solamente dalla RAM) il fatto che il programma stia girando a velocità normale non genera ambiguità su questi LED (cioé non è acceso più di un gruppo LED contemporaneamente).

E) Osservate i LED dei dati. Per motivi che saranno più oltre chiariti, quando un programma gira, sembrano tutti accesi. In effetti sono utili solo quando si è nel modo hardware step.

F) Premete [RESET]. Con questa operazione il programma viene fermato e il controllo ritorna al monitor. Il display visualizza l'istruzione che stava per essere eseguita quando è stato premuto [RESET].

G) Il monitor del μLab vi impedisce di far partire un programma mentre state inserendo dei dati. Vediamolo con un esempio. Premete [FETCH ADRS] [0] [8] [0] [0] e quindi [1] [2]. Il punto decimale è acceso e indica che il μLab è nel modo inserimento dati.

H) Premete [RUN]. Essendo nel modo inserimento dati, il μLab non risponde a questo comando.

I) Supponiamo che abbiate deciso di non cambiare il dato e che vogliate solo lanciare il programma. Premete [DECR] per uscire dal modo inserimento dati e quindi premete [STORE /INCR] per ritornare all'indirizzo 0800. Alternativamente potete premere [FETCH ADRS] [0] [8] [0] [0]. Si noti che il punto decimale è ora spento.

ESPERIMENTO 4-2

(continuazione)

J) Premete [RUN]. Il programma sta ora girando.

K) Premete [RESET] per ridare il controllo al monitor.

RIASSUNTO

In questo esperimento sono stati presentati i tre modi per eseguire un programma: instruction step, hardware step e run. La funzione dei tasti può così essere descritta:

[INSTR STEP] fa sì che venga eseguita l'istruzione visualizzata sul display. Il μLab rientra quindi nel programma monitor.

[HDWR STEP] fa sì che il microprocessore salti all'indirizzo presente sul display e poi si fermi. Premere altre volte il tasto fa sì che venga letto il byte visualizzato sui LED del bus dei dati e che, quando è stata letta un'istruzione intera, questa venga eseguita. Il microprocessore si ferma finché non viene premuto ancora HDWR STEP (oppure finché non viene premuto RESET).

[RUN] fa sì che il microprocessore inizi ad eseguire istruzioni partendo dall'indirizzo visualizzato sul display, continuando poi ad eseguire le istruzioni in sequenza.

[RESET] fa sì che il μLab rientri nel modo «attesa comando». Questo tasto è utilizzato per fermare un programma che sta girando (tasto di RUN) o per uscire dal modo hardware step. Non vengono modificati né i contenuti della memoria né i registri.

Due sono le differenze significative tra i due modi di esecuzione «step» (cioè passo passo). In primo luogo, nel modo hardware step, il microprocessore viene fermato, il display è spento e i LED binari sulle linee degli indirizzi, dei dati e dello stato rispecchiano effettivamente la situazione presente sui diversi bus. Si ottiene così una informazione più completa di quella che si può ottenere dal display, ma la sua interpretazione è più complicata. I LED vengono a essere una esatta indicazione di quello che sta succedendo nell'hardware del μLab. Nel modo instruction-step invece il display è attivo e non ha invece significato quanto è indicato dai LED dei dati e degli indirizzi.

La seconda differenza sta nel fatto che ogni qual volta si preme HDWR STEP viene prelevato dalla memoria di programma un solo byte, qualunque sia il numero di byte che compongono l'istruzione. L'istruzione quindi è eseguita solo quando tutte le sue parti (byte) sono state passo passo prelevate. D'altro canto, ogni qual volta che viene premuto il tasto INSTR STEP, viene eseguita l'istruzione visualizzata sul display. Se l'istruzione è composta da più byte, vengono prelevati anche il secondo e il terzo byte, che non sono mai mostrati sul display, e viene eseguita l'intera istruzione. Tale modo è perciò più utile per poter seguire il comportamento di un programma, mentre con il modo hardware step si può ottenere una descrizione più dettagliata di quanto viene effettuato da ogni istruzione.

Si osservi che in realtà il tasto HDWR STEP ha due funzioni. Quando il μLab non si trova ancora nel modo hardware step, premere una prima volta questo tasto vuol dire far entrare il sistema in tale modo. Solo ora che il μLab è già in questo modo, la pressione del tasto provoca l'avanzamento di un passo. Per ritornare all'operatività normale si usa il tasto di RESET.

Quando ci si trova nel modo RUN è difficile ricavare numerose informazioni su quanto sta accadendo. Questo modo perciò è utilizzato per provare programmi che eseguano una qualche operazione estremamente osservabile, a differenza del programma «fannullone» che abbiamo utilizzato.

LE PORTE D'INGRESSO E D'USCITA

Il μLab dispone di una porta d'ingresso cui sono collegati otto microinterruttori a slitta che definiscono lo stato degli ingressi. Il microprocessore può così leggere i dati dagli interruttori. Il μLab dispone anche di una porta d'uscita, ad ogni linea della quale (per un totale di otto linee) è collegato un LED; in questo modo i LED possono essere controllati dal microprocessore. Sono queste le porte che sono state utilizzate per l'esperimento sul dispositivo AND nella Lezione 3.

Figura 4-3. Diagramma di flusso del programma che copia i dati della porta d'ingresso su quella d'uscita.

La Figura 4-3 presenta il diagramma di flusso di un programma che mostra come si possono utilizzare queste porte. Il programma legge un dato dalla porta d'ingresso e lo scrive sulla porta d'uscita. Nella Tabella 4-2 è riportato il listing del programma.

Indirizzo	Contenuto	Label	Istruzione	Commenti
0900	3A	START:	LDA 2000	; Leggi la porta d'ingresso
0901	00			
0902	20			
0903	32		STA 3000	; Scrivi il dato
0904	00			sulla porta d'uscita
0905	30			
0906	C3		JMP START	; Ritorna all'inizio
0907	00			
0908	09			

Tabella 4-2. Programma che copia i dati dalla porta d'ingresso alla porta d'uscita

La prima istruzione è LDA 2000 il cui significato è «carica l'accumulatore con il contenuto della locazione 2000». Le porte d'ingresso e d'uscita sono indirizzate esattamente come se fossero locazioni di memoria; in particolare l'indirizzo della porta d'ingresso del μLab (vedi la Mappa di Memoria in Fig. 4-1) è pari a 2000. Questa istruzione quindi effettua l'operazione di caricare l'accumulatore con il dato prelevato dalla porta d'ingresso.

L'istruzione successiva, STA 3000, ha il significato di «carica il contenuto dell'accumulatore nell'indirizzo 3000», dove 3000 è l'indirizzo della porta d'uscita. Questa istruzione perciò prende il contenuto dell'accumulatore, che nel nostro caso corrisponde a quanto è stato letto sulla porta d'ingresso, e lo invia alla porta d'uscita.

L'ultima istruzione è una istruzione di salto. Tale istruzione fa sì che il programma salti indietro all'inizio in modo da effettuare un loop (cioè un anello) che può essere eseguito indefinitamente. Mentre il programma gira, quindi, il dato sulla porta d'uscita seguirà costantemente il dato sulla porta d'ingresso.

Tutte e tre queste istruzioni hanno lo stesso formato di codice macchina: un codice operativo seguito da due byte di informazione di indirizzo. Per esempio «3A» è il codice operativo della istruzione LDA, mentre i due byte successivi, 00 e 20, definiscono l'indirizzo 2000.

ESPERIMENTO 4-3

Uso delle porte d'ingresso e d'uscita

INTRODUZIONE

Viene inserito e fatto girare il programma, appena descritto, che trasferisce un dato dalla porta d'ingresso alla porta di uscita. Qualunque sia la configurazione in cui si disporranno gli interruttori della porta d'ingresso, sui LED della porta di uscita dovrà comparire lo stesso dato.

PROCEDIMENTO

A) Premete [FETCH ADRS] [0] [9] [0] [0].

B) Premete [3] [A] [STORE /INCR]. Viene memorizzato il primo byte del programma mostrato nella tabella 4-2.

C) Premete [0] [STORE /INCR]. Viene memorizzato il secondo byte del programma, 00, nella locazione 0901.

D) Continuate a memorizzare i dati della tabella 4-2 fino a che l'intero programma è stato introdotto.

E) Verificate che il programma introdotto è corretto.

F) Fate partire il programma premendo i tasti [FETCH ADRS] [0] [9] [0] [0] [RUN].

G) Disponete a piacere gli interruttori della porta d'ingresso. Si ricordi che l'alto corrisponde ad un uno, il basso a uno zero.

H) Osservate i LED della porta d'uscita: indicano lo stesso dato presente sugli interruttori della porta d'ingresso. Si noti che i LED della porta d'uscita operano in logica negativa, ovvero sono accesi ad indicare uno zero e sono spenti ad indicare un uno. Il motivo di questo sta nel fatto che il dispositivo TTL che li pilota non può fornire molta corrente, mentre è in grado di assorbirne una buona quantità. In genere i LED sono collegati appunto in questo modo.

I) Modificate gli interruttori della porta d'ingresso. In corrispondenza si dovrebbero vedere cambiare anche i dati sulla porta d'uscita.

J) Premete [RESET]. Il programma viene fermato e il controllo è riportato al monitor.

K) Modificate gli interruttori della porta d'ingresso. Cambiano i dati d'uscita? Poiché il programma che avete inserito non sta girando, la porta d'ingresso non è più letta.

(continuazione)

L) L'istruzione CMA (Complementa l'accumulatore, codice 2F) fa sì che venga complementato (invertito) il contenuto dell'accumulatore. Ogni bit a uno viene, dopo l'esecuzione di CMA, posto a zero e ogni bit a zero viene posto a uno. Inserite questa istruzione nel programma (al posto giusto!), cosicchè i dati letti dalla porta d'ingresso vengono complementati prima di essere inviati alla porta d'uscita. Per fare spazio alla nuova istruzione, ricordatevi di spostare opportunamente le istruzioni già presenti. Notate che ogni istruzione del programma era lunga 3 byte e che questi tre byte sono un tutt'uno e non possono essere spezzati. (Potete trovare la risposta corretta nella Tabella 4-3, al termine dell'esperimento).

M) Fate girare il programma così modificato e osservate se si comporta come previsto. Stavolta a un interruttore "alto" dovrebbe corrispondere un LED *acceso*.

N) Premete [RESET] a conclusione del programma.

RIASSUNTO

È stato letto un dato dalla porta d'ingresso e lo stesso è stato inviato alla porta d'uscita. Finché il programma ha girato, il dato visualizzato sui LED della porta d'uscita ha seguito lo stato degli interruttori della porta d'ingresso. Poiché i LED della porta d'uscita danno una indicazione in logica negativa, un interruttore *alto* corrisponde a un LED *spento*. Per modificare questa situazione, si è aggiunta una istruzione che complementa il contenuto dell'accumulatore. È questo un esempio della flessibilità di un sistema basato su microprocessore. Se gli interruttori fossero stati direttamente collegati ai LED, per fare tale modifica sarebbe stato necessario aggiungere otto dispositivi per invertire i segnali (invertitori, inverter). Dato che questo collegamento è effettuato attraverso un microprocessore, è stato invece sufficiente inserire una piccola modifica al software per ottenere il risultato richiesto.

Indirizzo	Contenuto	Label	Istruzione	Commenti
0900	3A	START:	LDA 2000	;Leggi la porta d'ingresso
0901	00			
0902	20			
0903	2F		CMA	;Complementa il dato
0904	32		STA 3000	;Scrivi il dato nella porta d'uscita
0905	00			
0906	30			
0907	C3		JMP START	;Ritorna all'inizio
0908	00			
0909	09			

Tabella 4-3. Programma modificato che complementa il dato

Il μLab dispone di tasti che con estrema facilità vi permettono di esaminare e modificare i contenuti della memoria, di far girare un programma o di farlo avanzare passo passo, in due diversi modi. Il modo instruction step vi permette, in particolare, di seguire come procede il programma, mentre il modo hardware step vi permette di vedere in dettaglio quanto si verifica sui bus dei dati e degli indirizzi.
Ricordatevi che le funzioni così associate ai tasti, sono definite dal programma monitor presente nella ROM e dai circuiti del μLab, e non dal microprocessore. La funzione delle singole istruzioni espresse in linguaggio macchina dipende invece proprio dal microprocessore.

DOMANDE

Lezione 4

1. Per esaminare il contenuto della locazione di memoria 0803, dovete premere _____0803.

2. Supponete che il display stia visualizzando 0803. Per vedere il contenuto della locazione 0802 dovete premere_____oppure_____.

3. Tra il modo hardware step e il modo istruction step, una delle differenze che si hanno, è che:
   a. l'hardware step vi permette di osservare lo stato dei bus.
   b. l'hardware step vi permette di seguire il programma sul display.
   c. l'instruction step avanza una locazione di memoria per volta.
   d. l'instruction step fa girare il programma a velocità normale.

4. Le porte d'ingresso sono identificate da_____, esattamente come le locazioni di memoria.

5. Le funzioni dei tasti del μLab sono così definite a causa di:
   a. il progetto del microprocessore.
   b. il programma monitor memorizzato nella ROM.
   c. il contenuto della RAM.
   d. tutto quanto detto nei punti a, b, c.

LEZIONE 5

Alcuni concetti di software

INTRODUZIONE

Nella lezione precedente sono state descritte alcune funzioni fondamentali del μLab. Servendosi dei tasti descritti in tale lezione è possibile memorizzare un programma, verificarlo, eseguirlo un passo per volta oppure farlo girare normalmente. Queste sono le caratteristiche essenziali del μLab. In questa lezione verranno descritte alcune tecniche di programmazione e alcune caratteristiche più sofisticate del μLab.

I REGISTRI DEL MICROPROCESSORE

Il microprocessore 8085, un solo circuito integrato, contiene un certo numero di registri interni che sono variamente utilizzati. Alcuni, ad esempio l'accumulatore, sono utilizzati per memorizzare e manipolare dei dati. Ad esempio nella Lezione 3 l'accumulatore è stato utilizzato per memorizzare il numero dei conteggi nel programma "Conta a dieci". Il contenuto dell'accumulatore può anche essere modificato (nell'esempio citato, era incrementato). Questi registri si differenziano dalle solite locazioni di memoria per il fatto che sono interni al microprocessore e non sono selezionati attraverso il bus degli indirizzi. (Fig. 5-1)

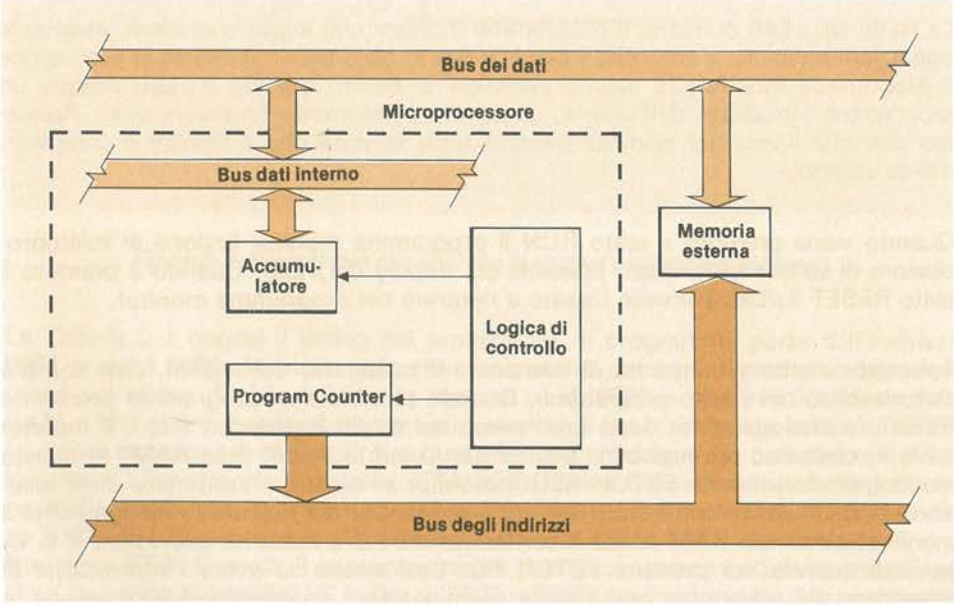


Figura 5-1. Registri e Memoria. I registri, ad esempio l'accumulatore, sono direttamente selezionati dalla logica di controllo, mentre alle locazioni di memoria l'accesso è reso possibile attraverso i bus dei dati e degli indirizzi.

In effetti sono direttamente selezionati solo da alcune istruzioni particolari (p.es., MVI A seleziona l'accumulatore). È la logica di controllo interna al microprocessore che controlla direttamente tali registri, senza utilizzare i bus esterni. Per questo motivo sono utili per memorizzare dei dati provvisori o parziali.

Alcuni dei registri del microprocessore non sono genericamente utilizzati per immagazzinare dati, ma sono dedicati a funzioni assai specifiche. Il più importante tra questi registri specializzati è il *Program Counter* (PC, Contatore di Programma). È un registro a 16 bit il cui compito è di indicare quale istruzione deve essere eseguita: contiene sempre l'indirizzo della prossima istruzione da eseguire. Quando deve essere letta una istruzione dalla memoria, sul bus degli indirizzi viene posto il contenuto del program counter. L'istruzione così indirizzata è quindi presentata sul bus dei dati; il microprocessore legge tale istruzione e quindi incrementa il program counter. Il contenuto del PC è di nuovo posto sul bus degli indirizzi e viene letta l'istruzione successiva. Questo processo continua finché non viene incontrata una istruzione di salto. In tal caso l'istruzione di salto fa sì che il PC venga caricato con l'indirizzo del salto.

Gli altri registri presenti nel microprocessore verranno introdotti più avanti nel corso del libro.

SEGUIRE IL FLUSSO DI UN PROGRAMMA

Per seguire passo passo con il μLab, come opera un programma, è utile poter conoscere il contenuto dei registri del microprocessore. Poiché questi sono interni al microprocessore e non hanno indirizzo per accedervi, non si può utilizzare il tasto FETCH ADRS. A tal fine si usa un tasto speciale, FETCH REG (Fetch Register, preleva il registro). Un altro tasto, FETCH PC (Fetch Program Counter, preleva il contatore di programma) vi permette di ritornare con facilità a un programma che è stato interrotto.

IL PROGRAMMA MONITOR DEL μLAB

La ROM del μLab contiene il programma monitor che legge la tastiera, esegue le operazioni indicate, e controlla il display. Sul μLab è continuamente in esecuzione il programma monitor; le uniche eccezioni si hanno quando il μLab esegue un programma introdotto dall'utente, o quando è nel modo hardware step. Possiamo dire che il monitor non sta girando tutte le volte che il display è completamente spento.

Quando viene premuto il tasto RUN il programma monitor impone al microprocessore di saltare all'indirizzo presente sul display del μLab. Quando è premuto il tasto RESET il μLab è invece forzato a rientrare nel programma monitor.

Il monitor inoltre vi permette di esaminare il contenuto dei registri, così come è stato lasciato dal vostro programma. Quando terminate un programma premendo RESET (e analogamente dopo ogni passo nel modo instruction step), il monitor salva il contenuto dei registri in alcune particolari locazioni della RAM. In questo modo quando premete FETCH REG il monitor vi mostra il contenuto della locazione di memoria in cui è stato salvato il contenuto del registro. Analogamente il monitor salva nella RAM anche il contenuto del PC e richiama tale valore e lo visualizza quando voi premete FETCH PC. Così anche se avete l'impressione di esaminare dei registri, in realtà state esaminando il contenuto delle memorie in cui il monitor ha memorizzato e salvato il contenuto dei registri. Questo modo di operare è reso necessario dal fatto che il monitor stesso utilizza i registri del microprocessore.

UN PROGRAMMA CONTATORE

Per mostrare come si possono utilizzare questi tasti, ci serve d'esempio un altro programma. La Figura 5-2 riporta il diagramma di flusso di un programmatore "contatore" che fa sì che la porta d'uscita conti (in binario) da 0 a 255, riprendendo poi il conteggio dal valore 0. Per prima cosa vengono posti a 0 alcuni registri (in questo caso l'accumulatore, registro A). Il contenuto dell'accumulatore è quindi riportato sulla porta d'uscita e l'accumulatore stesso incrementato. Ancora, da ultimo, il programma salta indietro al punto in cui viene effettuato il trasferimento verso l'uscita.

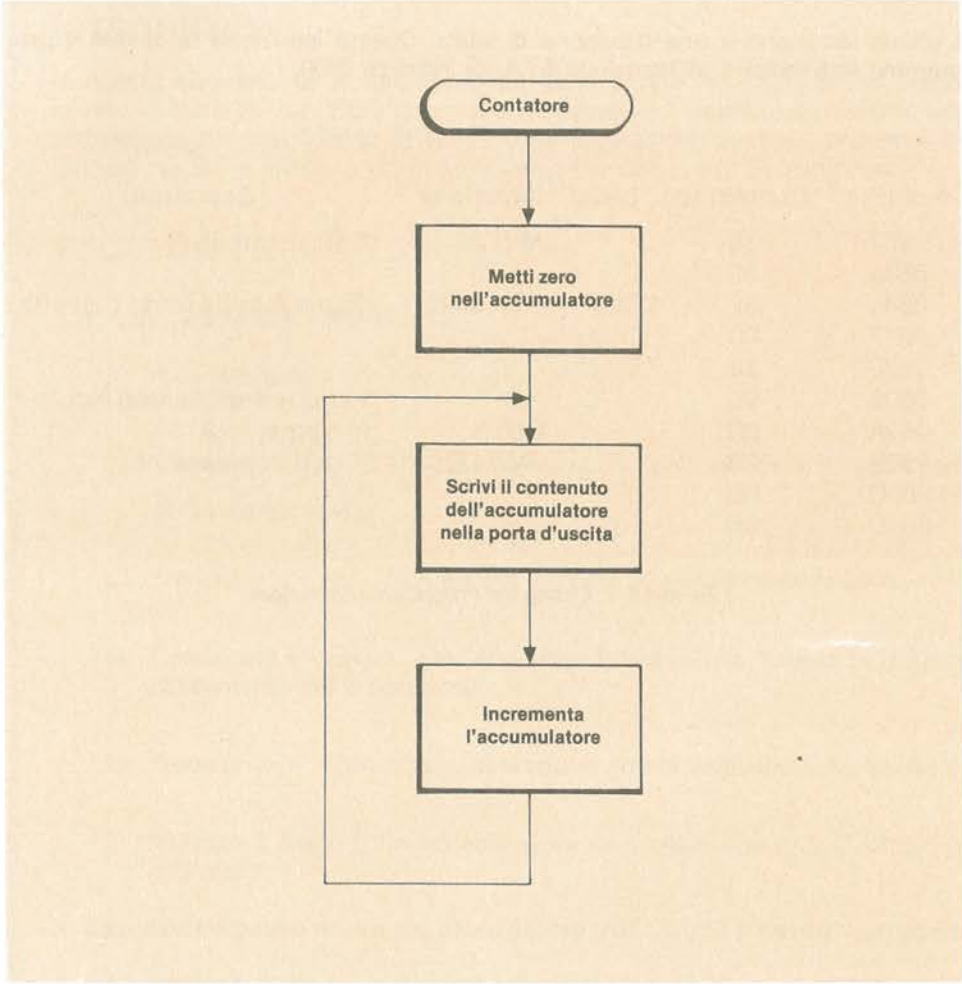


Figura 5-2. Programma che fa sì che la porta d'uscita conti in binario.

La Tabella 5-1 riporta il listing del programma. Il programma parte dall'indirizzo 0804 anziché 0800, in modo tale che voi potrete poi aggiungere altre istruzioni prima del punto d'inizio (si ricorda che nel μLab 0800 è la prima locazione della RAM e che quindi l'utilizzatore non può servirsi di locazioni poste prima di 0800). La prima istruzione è MVI A,0 che fa sì che l'accumulatore sia caricato con il valore zero. L'istruzione successiva, STA 3000 (carica l'accumulatore nella locazione 3000), trasferisce il contenuto dell'accumulatore nella porta d'uscita (di indirizzo 3000). Il codice operativo 32 posto all'indirizzo 0806 indica che si tratta di una istruzione STA. Quando il microprocessore riconosce questo codice operativo, sa che i due byte successivi (di indirizzo 0807 e 0808) contengono l'indirizzo della locazione in cui va memorizzato il contenuto dell'accumulatore (in questo caso 3000). Si noti che questa istruzione non modifica il contenuto dell'accumulatore, ma che si limita a copiare il dato nella porta d'uscita.

L'istruzione STA è eseguita da un NOP che serve solo a lasciare un byte di spazio che potremo utilizzare. L'istruzione successiva è INR A che incrementa il contenuto dell'accumulatore. Quando l'accumulatore raggiunge il massimo valore di conteggio (11111111 in binario, FF in esadecimale, 255 in decimale), il conteggio riprende da capo e al prossimo incremento l'accumulatore conterrà zero. Questo comportamento è identico a quello che si ha in un circuito integrato contatore binario standard, per esempio il 74163. Un particolare flip-flop, interno al microprocessore, detto *flag di riporto* (carry), viene posto a uno, ad indicare che il conteggio ha superato il massimo consentito. Nella lezione 12 verrà discusso più in dettaglio il flag di riporto.

L'ultima istruzione è una istruzione di salto. Questa istruzione fa sì che il programma salti indietro all'istruzione STA, di indirizzo 0806.

Indirizzo	Contenuto	Label	Istruzione	Commenti
0804	3E		MVI A,0	;Carica zero in A
0805	00			
0806	32	LOOP:	STA 3000	;Copia A sulla porta d'uscita
0807	00			
0808	30			
0809	00		NOP	;Per una espansione futura
080A	3C		INR A	;Incrementa A
080B	C3		JMP LOOP	;Ripeti l'operazione
080C	06			
080D	08			

Tabella 5-1. Listing del Programma Contatore.

Esecuzione del programma contatore

INTRODUZIONE

In questo esperimento vi viene proposto di inserire e far girare passo passo il programma appena discusso. Il tasto FETCH REG (fetch register, preleva il registro) è utilizzato per esaminare i registri utilizzati dal programma. Il tasto FETCH PC (fetch program counter, preleva il contatore di programma) vi renderà più facile rientrare nel programma una volta che lo avete fermato.

PROCEDIMENTO

I. Introduzione del programma

A) Premete [FETCH ADRS] [0] [8] [0] [4]

B) Premete [3] [E] [STORE /INCR]. Con questa operazione caricate il codice operativo 3E (istruzione MVI A) nella locazione 0804.

C) Premete [0] [STORE /INCR]. Con questo caricate 00 nella locazione 0805.

D) Continuate a inserire i dati opportuni (riferitevi alla Tabella 5-1) finché non avete completato il caricamento del programma.

E) Premete [DECR]. Con questa operazione potete verificare la locazione precedente.

F) Ripetete il passo E finché non avete controllato che tutto il programma sia stato caricato correttamente.

II. Esecuzione passo passo del programma utilizzando il modo instruction step

A) Premete [FETCH ADRS] [0] [8] [0] [4] (se il display visualizza già 0804, non eseguite questa operazione). Sul display compare l'istruzione MVI A (codice operativo 3E)

B) Premete [INSTR STEP] per eseguire l'istruzione MVI A (codice operativo 3E).

C) Sul display compare ora il codice operativo dell'istruzione STA (32). Premete [INSTR STEP]. Che cosa è successo? L'istruzione ha inviato una configurazione di tutti zeri alla porta d'uscita. Operando in logica negativa, tutti i LED si sono accesi. Infatti ciascun LED si accende ad indicare uno zero, si spegne ad indicare un uno.

D) Premete [INSTR STEP] tre volte. Con questa operazione eseguite le istruzioni NOP, INR A e JMP.

ESPERIMENTO 5-1

(continuazione)

E) Sul display compare ancora il codice operativo STA (32). Il contenuto dell'accumulatore è stato però incrementato. Quando questa istruzione verrà eseguita, questo nuovo valore verrà presentato sui LED della porta d'uscita. Verificatelo premendo [INSTR STEP].

F) Premete ripetutamente [INSTR STEP] in modo da vedere la sequenza ripetersi. Ricordando che un LED spento equivale a uno e un LED acceso a uno zero, potete verificare che i LED della porta d'uscita stanno contando in binario.

G) Quando l'istruzione NOP (indirizzo 0809, codice operativo 00) compare sul display cessate di fare avanzare il programma. Premete [FETCH REG]. Potete così vedere il contenuto dei registri del processore. Il display visualizza un "A" e un altro dato. Il dato presentato è il valore che è contenuto nell'accumulatore (registro A) dopo l'ultimo passo del programma eseguito. Il dato corrisponde (convertito in binario) al numero che è visualizzato sui LED della porta d'uscita. Infatti a questo punto del programma il dato è appena stato inviato dall'accumulatore alla porta d'uscita e il display vi sta proprio mostrando il contenuto dell'accumulatore. (Ricordatevi, quando fate un confronto tra i LED e l'accumulatore, che i LED danno indicazioni in logica negativa).

H) Premete [STORE /INCR]. Il display vi mostra ora il registro dei flag (descritto nella Lezione 12).

I) Premete ripetutamente [STORE /INCR] finché il display visualizza PCH, la parte più significativa del program counter (08).

J) Premete [STORE /INCR]. Compare ora la parte meno significativa del PC (09). Nella Lezione 11 vi verranno descritti gli altri registri che avete visto visualizzati prima che comparisse PC.

K) Per riprendere l'esecuzione del programma premete [FETCH PC]. Con questa operazione richiamate l'ultimo valore del program counter, l'indirizzo cioè della prossima istruzione da eseguire. Il display vi mostra ora tale indirizzo (0809). Come vi è stato descritto in questa lezione il programma monitor salva (in memoria) il valore del program counter dopo l'esecuzione di ogni passo. Con il tasto [FETCH PC] recuperate appunto questo valore.

L) Premete [INSTR STEP]. Con questa operazione avete ripreso a procedere passo passo all'interno del programma.

M) Continuate a procedere nella sequenza del programma (utilizzando sempre [INSTR STEP]) e fermatevi quando siete ritornati allo stesso punto di prima (indirizzo 0809). Premete [FETCH REG]. Il valore dell'accumulatore si è incrementato e lo stesso valore è riportato sui LED della porta d'uscita.

N) Premete [FETCH PC] e ripetete il passo M verificando che il valore dell'accumulatore venga ulteriormente incrementato.

III. Esecuzione passo passo del programma utilizzando il modo hardware step

A) Potete eseguire passo passo il programma ed esaminare i registri anche utilizzando [HDWR STEP] al posto di [INSTR STEP]. Premete [FETCH ADRS] [0] [8] [0] [4] [HDWR STEP]. Poiché siete ora nel modo hardware step il display si è abbuiato. L'indirizzo 0804 compare ora in forma binaria sui LED degli indirizzi. Sui LED dei dati compare invece in binario il dato, il codice operativo 3E, corrispondente a tale locazione. 3E è il codice operativo dell'istruzione MVI A. Ricordatevi che quando utilizzate il modo hardware dovete sempre fare riferimento ai LED binari degli indirizzi e dei dati.

B) Premete [HDWR STEP]. Con questa operazione prelevate il secondo byte dell'istruzione, il byte 00.

C) Premete [HDWR STEP]. Viene così eseguita l'istruzione.

D) Sui LED dei dati compare ora il codice operativo di STA, (32). Premete [HDWR STEP]. Con questa operazione prelevate il secondo byte dell'istruzione, la parte bassa dell'indirizzo della porta (00).

E) Premete [HDWR STEP]. Sui LED dei dati compare ora la parte alta dell'indirizzo della porta (30).

F) Premete [HDWR STEP]. L'istruzione è stata completamente letta dalla memoria ed è necessario un ulteriore passo perché venga eseguita. I LED degli indirizzi visualizzano ora 3000, l'indirizzo della porta, e i LED dei dati 00, il dato che deve essere inviato alla porta.

G) Premete [HDWR STEP]. L'istruzione è stata ora eseguita e i LED della porta d'uscita sono infatti tutti accesi.

H) Il programma si trova a questo punto all'altezza dell'istruzione NOP (00), che è il punto in cui nella parte II di questo esperimento avete esaminato l'accumulatore. Premete [RESET] per riportarvi in monitor. L'istruzione NOP è stata eseguita e il display visualizza l'indirizzo della prossima istruzione (080A).

I) Premete [FETCH REG] per vedere il contenuto dell'accumulatore.

J) Per ritornare al programma premete [FETCH P.C.] [HDWR STEP].

K) Premete ripetutamente [HDWR STEP] per eseguire un'altra volta l'intera sequenza, fermatevi ancora allo stesso punto (indirizzo 0809).

L) Ripetete i passi H, I, J, e K più volte in modo da vedere l'accumulatore che si incrementa. Il valore contenuto nell'accumulatore corrisponde ogni volta al valore visualizzato sui LED della porta d'uscita.

M) Premete [RESET] per riportare il μLab al modo di funzionamento normale.

RIASSUNTO

In questo esperimento avete caricato in memoria il programma che faceva contare in modo binario i LED della porta d'uscita. Servendosi dei due modi di esecuzione passo passo avete poi potuto verificare il programma. Il tasto FETCH REG vi ha permesso di esaminare il contenuto dell'accumulatore mentre con il tasto FETCH PC avete potuto riportarvi all'indirizzo in cui avevate interrotto l'esecuzione passo passo del programma, potendo così riprendere l'esecuzione stessa. Quando vi trovate nel modo hardware step avete premuto poi il tasto RESET per riportarvi in monitor e per poter così utilizzare il tasto FETCH REG o gli altri tasti.

Notate che sempre nel modo hardware step l'istruzione STA ha richiesto, per essere eseguita, un passo supplementare. Per ogni riferimento alla memoria o alle porte di I / O è infatti necessario un passo (hardware step); per questo motivo l'istruzione STA vi ha richiesto tre passi per leggere i tre byte dell'istruzione e un quarto passo per scrivere nella porta d'uscita.

ORGANIZZAZIONE DEL PROGRAMMA

Nella maggior parte dei casi le applicazioni con i microprocessori sono costituite da un insieme di un certo numero di compiti (task) che devono essere svolti. Un voltmetro basato su microprocessore, per esempio, leggerà una tensione d'ingresso e invierà il dato corrispondente al display. In questo caso il processore può servire ad effettuare l'azzeramento automatico e a manipolare i dati per convertire le unità di misura. In uno strumento più sofisticato può anche essere presente una tastiera attraverso cui l'utente ha la facoltà di richiedere funzioni particolari che sarà compito del processore acquisire e interpretare. Ancora, per poter effettuare la scelta automatica della scala di misura (autoranging), il processore deve controllare un attenutatore. Tutti questi compiti sono fondamentalmente indipendenti l'uno dall'altro e a ciascuno di essi può quindi essere associato un programma particolare abbastanza semplice. Un programma principale, che non entra nei dettagli dei singoli compiti, avrà poi la funzione di coordinare tutti i programmi particolari.

La situazione descritta è analoga a quella che si verifica in una grande industria. Il presidente non può fare tutto: il suo compito è quello di prendere solo le decisioni importanti. Non è invece necessario che si preoccupi di quello che viene servito in mensa né del tipo di strofinacci che usano gli addetti alle pulizie: queste decisioni saranno demandate ad altre persone. Esisteranno poi alcuni vice presidenti ciascuno dei quali sarà responsabile di un settore particolare della ditta. Sotto di questi, ancora, lavoreranno dei dirigenti e dei capi ufficio cui spetterà finalmente il compito di controllare i vari impiegati e operai.

Tutti nella ditta hanno un compito da svolgere, ma le singole persone hanno tanto più a che fare con lavori di dettaglio quanto più bassa è la loro posizione all'interno dell'ordine gerarchico presente. In questo modo, il presidente ha tempo per prendere decisioni importanti perché può chiedere che altre persone seguano la loro esecuzione. Queste a loro volta possono appoggiarsi ad altre persone che se necessario possono servirsi di ulteriori persone. Ogni impiegato riferisce così al suo "superiore" e, quando opportuno, il presidente è infine in grado di ricevere una risposta.

Un sistema a microprocessore ha un modo di operare molto simile a quello appena descritto. I compiti, poiché possono essere del tutto diversi e comunque complicati, non vengono svolti di solito da un solo programma, ma vengono affidati a diversi programmi particolari. Un programma, che è spesso chiamato *executive* (esecutivo), ha la funzione di "presidente". L'executive dispone e controlla gli altri programmi che effettuano il lavoro in sua vece, i quali poi a loro volta possono chiamare ulteriori programmi a svolgere dei compiti specifici.

I programmi che lavorano per altri programmi sono detti *subroutine* (sottoprogrammi, spesso dette più semplicemente *routine*). Proprio come nel caso della ditta con diversi livelli gerarchici, le subroutine possono disporre a loro volta di altre subroutine che svolgono dei compiti in loro vece.

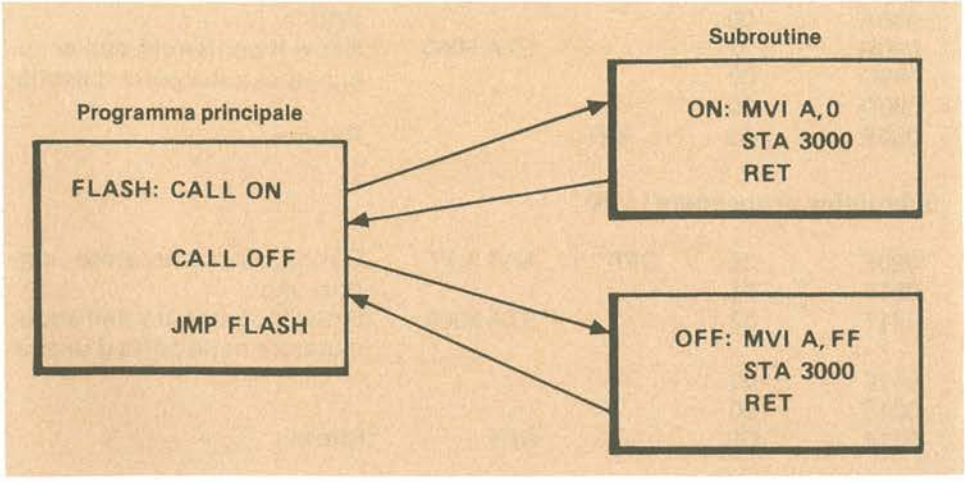


Figura 5-3. Utilizzo di subroutine per far lampeggiare (Flash) i LED della porta d'uscita

UN ESEMPIO

Vediamo ora di scrivere un programma, che utilizzi delle subroutine, per accendere e spegnere in continuazione i LED della porta d'uscita. Una soubroutine (ON) sia specializzata ad accendere i LED, l'altra (OFF) a spegnerli (Fig. 5-3).

Il programma principale è costituito da tre sole istruzioni: una chiamata dalla routine ON, una chiamata dalla routine OFF e un salto indietro al punto di partenza. Il programma principale definisce la funzione complessiva che deve essere svolta ma non si interessa del fatto che i LED della porta di uscita operano in logica negativa, né vuol sapere a quale indirizzo si trova la porta di uscita stessa. Questi dettagli vengono curati dalle sobroutine particolari.

L'istruzione *CALL* (chiama) fa sì che il microprocessore effettui un salto ad una subroutine. La label (etichetta) ON, che identifica la subroutine ON che si vuole chiamare, viene sostituita, durante l'operazione di traduzione da linguaggio di assemblaggio a codice macchina, con un indirizzo.

Il programma completo è riportato nella Tabella 5-2 in cui è riportato pure il codice macchina. L'istruzione CALL è indicata dal codice operativo CD che è immediatamente seguito dall'indirizzo della subroutine memorizzato esattamente come se fosse un indirizzo di salto (istruzioni di JUMP). L'istruzione RET è costituita soltanto dal codice operativo C9 e non comprende alcun indirizzo.

Programma principale

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	CD	FLASH:	CALL ON	;Accendi i LED
0801	09			
0802	08			
0803	CD		CALL OFF	;Spegni i LED
0804	0F			
0805	08			
0806	C3		JMP FLASH	;Ripeti
0807	00			
0808	08			
Subroutine per accendere i LED				
0809	3E	ON:	MVI A,00	;Carica l'accumulatore con zero
080A	00			
080B	32		STA 3000	;Scrivi il contenuto dell'accumulatore sulla porta d'uscita
080C	00			
080D	30			
080E	C9	RET		;Ritorno
Subroutine pr spegnere i LED				
080F	3E	OFF:	MVI A,FF	;Carica l'accumulatore con tutti uno
0810	FF			
0811	32		STA 3000	;Scrivi il contenuto dell'accumulatore nella porta d'uscita
0812	00			
0813	30			
0814	C9		RET	;Ritorno

Tabella 5-2. Programma che fa lampeggiare i LED utilizzando delle subroutine

Dovreste già conoscere le prime due istruzioni presenti nelle subroutine. L'ultima istruzione, *RET* (Retourn, ritorno), indica che la subroutine è terminata e fa sì che il microprocessore ritorni di conseguenza al programma principale.
Vi può stupire il fatto che il microprocessore sappia dove ritornare quando, al termine di una subroutine, incontra una istruzione di ritorno. In effetti quendo viene eseguita una istruzione CALL, viene memorizzato in sezione particolare della memoria detta *stack* (pila, catasta) l'indirizzo di ritorno, l'indirizzo cioè dell'istruzione che segue l'istruzione di chiamata di subroutine. Quando poi, al termine di una subroutine, incontra una istruzione di ritorno, il microprocessore si riprende dallo stack l'indirizzo di ritorno e può così continuare l'esecuzione del programma principale. Particolari più completi su come opera lo *stack* e sul perché è utilizzato, vi verranno dati nella Lezione 12. Fortunatamente tale operazione è quasi del tutto automatica.

ESPERIMENTO 5-2

Subroutine

INTRODUZIONE

In questo esperimento inserirete nel μLab ed eseguirete il programma, appena descritto, che fa lampeggiare i LED. Osserverete in particolare il flusso del programma che passa dal programma principale alle subroutine e viceversa.

PROCEDIMENTO

A) Introducete da tastiera il programma riportato nella Tabella 5-2.

B) Verificate che il programma sia stato correttamente memorizzato.

C) Premete [FETCH ADRS] [0] [8] [0] [0] . Il display visualizza ora la prima istruzione CALL (codice operativo CD).

D) Premete [INSTR STEP] . Viene eseguita l'istruzione CALL e l'indirizzo salta al valore 0809 (il punto di inizio della subroutine ON).

E) Premete [INSTR STEP] [INSTR STEP] . Si accendono i LED della porta d'uscita e sul display appare l'istruzione RET (indirizzo 080E, codice operativo C9).

F) Premete [INSTR STEP] . Viene effettuato il ritorno e l'indirizzo ritorna al programma principale. Sul display viene ora visualizzata la seconda istruzione CALL (indirizzo 0803, codice operativo CD).

G) Premete [INSTR STEP] . Viene eseguita l'istruzione CALL e il display visualizza ora l'indirizzo 080F (punto di inizio della subroutine OFF).

H) Premete [INSTR STEP] [INSTR STEP] . I LED vengono spenti e sul display compare l'istruzione RET (indirizzo 0814, codice operativo C9).

I) Premete [INSTR STEP] . Viene effettuato il ritorno e il display visualizza ora l'ultima istruzione del programma principale (indirizzo 0806, codice operativo C3).

J) Premete [INSTR STEP] . Il programma ritorna al punto di partenza (0800).

K) Premete [INSTR STEP] più volte in modo da vedere il programma ripetersi.

RIASSUNTO

Avrete inserito nel μLab le tre parti del programma che fa lampeggiare i LED: il programma principale, la subroutine che fa accendere i LED (ON) e quella che li spegne (OFF). Eseguendo passo passo il programma principale avrete osservato che l'istruzione CALL provoca un salto a una subroutine, mentre l'istruzione RET fa sì che il flusso del programma ritorni al programma principale. Anche se l'istruzione di ritorno non specifica un indirizzo, il microprocessore è in grado di effettuare un ritorno all'indirizzo dell'istruzione che segue il CALL (chiamata) precedente della subroutine.

INTERRUZIONI

Può capitare che il sistema a microprocessore debba gestire eventi poco frequenti e non prevedibili. Riprendendo ancora una volta l'esempio precedente, una ditta può aspettarsi che si verifichino certe situazioni, ma può pur tuttavia non conoscere il momento preciso in cui queste si presenteranno. Ad esempio si può verificare un incendio, tutti interrompono il lavoro e vengono attuate le misure di sicurezza previste. Quando il fuoco è stato spento e sono stati riparati i danni, il personale può ritornare al lavoro e riprendere così le solite operazioni.

Una situazione del tutto analoga si può verificare in un sistema a microprocessore. Prendiamo per esempio un voltmetro basato su microprocessore che disponga di un pulsante di calibrazione inserito nel pannello. Quando tale pulsante viene premuto, il microprocessore cessa di eseguire l'operazione corrente, qualunque essa sia, e salta una subroutine. Quando l'operazione di calibrazione è stata completata, il controllo ritorna al programma che era stato interrotto.

Questi tipi di eventi sono detti *interruzioni* (interrupt). Il chip del microprocessore dispone di un ingresso attraverso cui può essere generata una interruzione: un livello alto su tale piedino fa sì che il microprocessore interrompa l'operazione in corso e salti a una speciale subroutine di interruzione. Questa subroutine, a sua volta, svolgerà il compito richiesto dal dispositivo che ha generato l'interruzione. Una istruzione di ritorno farà sì, al termine della routine di interruzione, che venga ripreso il programma che era stato interrotto. Nel microprocessore 8085 sono presenti cinque ingressi di interruzioni a cui corrispondono diversi tipi di interruzione, per la cui descrizione si rinvia alla Lezione 10.

Sul μLab il tasto INTRPT è collegato a uno di questi ingressi d'interruzione. Quando premete questo tasto, il microprocessore effettua un salto, qualunque cosa stesse facendo, ad una certa locazione in RAM, dove è memorizzata la subroutine di interruzione (detta anche *routine di servizio dell'interruzione*). Poiché in certe occasioni vi può capitare di voler ignorare le interruzioni, potete in tal caso disabilitarle dal programma. Così per esempio l'interruzione generata dal tasto INTRPT è normalmente disabilitata dal programma monitor, cosicché non rischiate di effettuare un salto alla RAM se non avete già introdotto un programma di servizio dell'interruzione.

L'USO DEL TASTO INTERRUPT DEL μLAB

Per esemplificare questo processo si può interrompere il program counter mostrato all'inizio di questa lezione attivando il tasto INTRPT.
Per ottenere questo è necessario aggiungere, all'inizio del programma, alcune istruzioni che abilitino l'ingresso d'interruzione. Nella Tabella 5-3 è riportato il listing della routine di abilitazione della interruzione.

Indirizzo	Contenuti	Istruzioni	Commenti
0800	3E	MVI A, 0D	;Carica nell'accumulatore il valore della maschera d'interruzione
0801	0D		
0802	30	SIM	;Copia l'accumulatore nella maschera d'interruzione
0803	FB	EI	;Abilita le interruzioni

Tabella 5-3. Routine di abilitazione dell'interruzione

La *maschera d'interruzione* è un particolare registro presente nel microprocessore che indica quali interruzioni dovrebbero essere abilitate e quali no (una descrizione più dettagliata è data nell'Appendice B, dove si descrive l'istruzione SIM). Per il momento, vi basti sapere che, quando nel registro maschera d'interruzione è inserito il valore esadecimale 0D, viene abilitato il tasto INTRTP. Per prima cosa il programma mette tale valore nell'accumulatore (è la solita istruzione MVI A). Quindi l'istruzione SIM (Set Interrupt Mask, definisci la maschera d'interruzione) trasferisce il contenuto dell'accumulatore nel registro maschera d'interruzione. Il contenuto di quest'ultimo registro indica al processore quali dovrebbero essere le interruzioni abilitate. L'istruzione successiva EI (Enable Interrupts, abilita le interruzioni) fa sì che esse vengano effettivamente abilitate.

Ora che le interruzioni sono state abilitate, è necessaria una routine di servizio dell'interruzione. Il monitor contiene una routine che attraverso l'altoparlante presente sul μLab genera un suono d'avvertimento. La Tabella 5-4 riporta il listing della routine di interruzione che chiama appunto la subroutine beep (posta all'indirizzo 0010).

Indirizzi	Contenuti	Istruzioni	Commenti
0AFC	CD	CALL BEEP	;Salta al programma beep
0AFD	10		
0AFE	00		;Abilita di nuovo
0AFF	FB	EI	le interruzioni
0B00	C9	RET	;Ritorna al programma contatore

Tabella 5-4. Routine di servizio dell'interruzione

La subroutine beep fa sì che l'altoparlante generi un suono di circa 1 KHz di frequenza: un "bip" (beep). Poiché la routine beep termina con un'istruzione di ritorno, il programma ritorna, dopo un breve suono, alla routine di servizio dell'interruzione. La Figura 5-4 mostra la sequenza di eventi che si verificano quando si ha un'interruzione. La Figura 5-5 riporta invece il diagramma di flusso dettagliato.

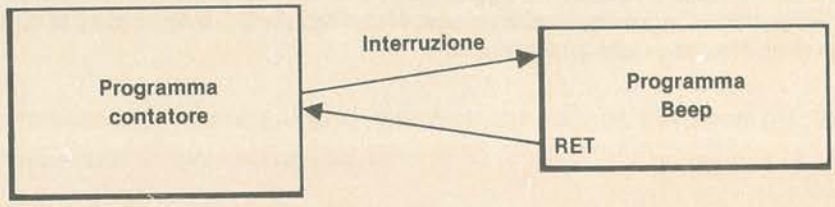


Figura 5-4. Sequenza semplificata degli eventi che si verificano quando il programma contatore viene interrotto

È necessario inserire nel programma l'istruzione EI poiché il microprocessore quando ha riconosciuto un'istruzione e ha effettuato il salto alla routine di servizio dell'interruzione, disabilita automaticamente le interruzioni stesse. Se non fosse inserita l'istruzione EI, verrebbe generato un «beep» solo la prima volta che avete premuto INTRTP, ma da tale momento in poi l'interruzione rimarrebbe disabilitata.
La routine di servizio dell'interruzione termina con un'istruzione di ritorno, cosicché, quando è stato generato il bip di avvertimento, il controllo ritorna al programma che era stato interrotto.

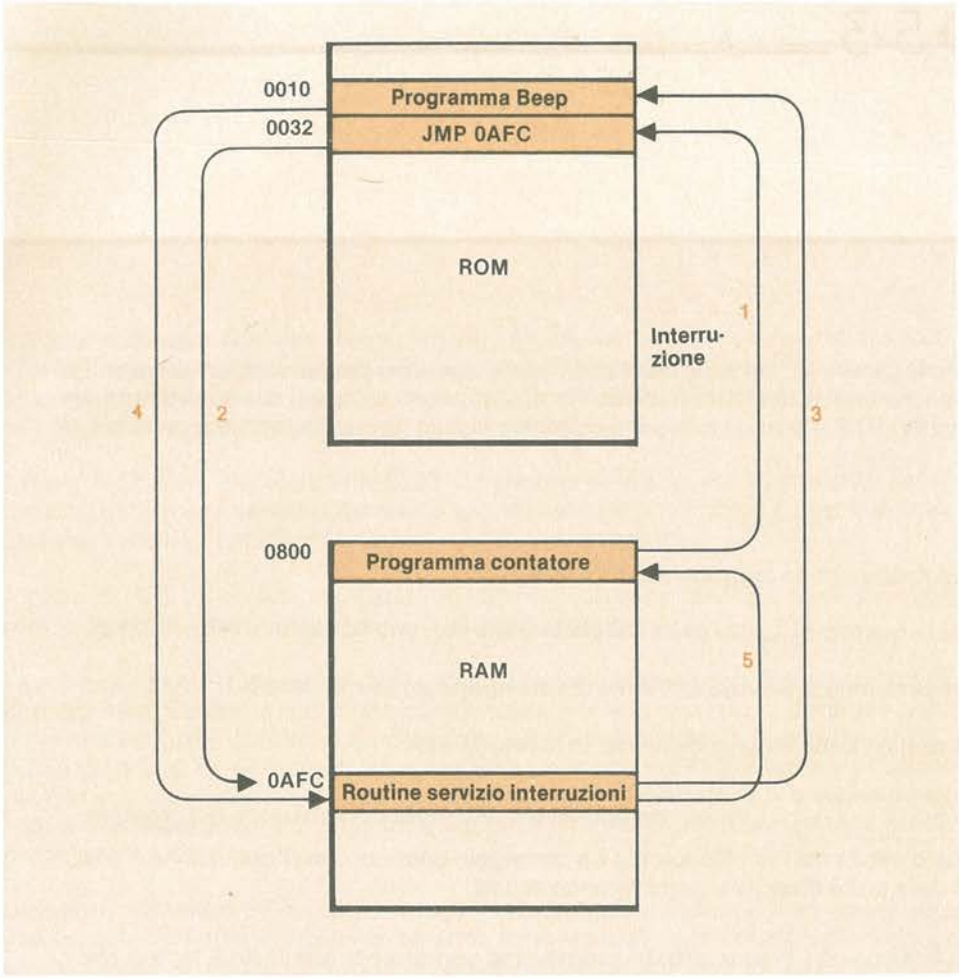


Figura 5-5. Flusso dettagliato del programma quando il programma contatore è interrotto

L'indirizzo 0AFC (il punto di partenza della routine di servizio della interruzione) non è un valore arbitrario come le locazioni degli altri programmi utente. Il tasto INTRTP fa sì che il microprocessore salti all'indirizzo 0034, che è una locazione fissa, non modificabile, così definita dal progetto stesso del microprocessore 8085. Tale locazione (che nel μLab è una locazione ROM) contiene l'istruzione JMP 0AFC; l'indirizzo 0AFC coincide con la locazione RAM in cui deve essere memorizzata la routine di servizio della interruzione. Potete comunque verificare che l'indirizzo 0034 contiene l'istruzione di salto a 0AFC: vi basta andare a vedere il contenuto delle locazioni 0034, 0035 e 0036.

ESPERIMENTO 5-3

Interruzioni

INTRODUZIONE

Viene fatto girare il programma contatore, cui sono state aggiunte le istruzioni che servono ad abilitare il tasto INTRTP. Viene inserita anche una routine di servizio dell'interruttore che chiama a sua volta il programma BEEP. Premendo il tasto INTRTP è così possibile interrompere il programma contatore e generare un "bip" di avvertimento.

PROCEDIMENTO

A) Introducete da tastiera il programma riportato nella Tabella 5-1.

B) Introducete da tastiera le quattro istruzioni della Tabella 5-3 che servono ad abilitare le interruzioni.

C) Introducete da tastiera la routine di servizio dell'interruzione riportata nella Tabella 5-1.

D) Verificate che tutte le routine siano state correttamente memorizzate.

E) Premete [FETCH ADRS] [0] [8] [0] [0] [RUN] . Con queste operazioni fate partire l'esecuzione del programma contatore. La porta d'uscita sta ora effettuando un conteggio continuo, ma l'operazione è così veloce che tutti i LED della porta d'uscita vi sembreranno accesi.

F) Premete [INTRPT] . Il μLab interrompe l'esecuzione del programma contatore e, per tutto il tempo che tenete premuto il tasto [INTRPT], esegue la routine Beep. Poiché il programma contatore non sta girando (è in esecuzione la routine di servizio dell'interruzione), i LED della porta d'uscita rimangono nello stato in cui si trovavano quando è stata generata l'interruzione. Quando rilasciate [INTRPT], il programma contatore riprende a girare.

G) Premete un'altra volta il tasto [INTRPT]. Poiché il programma contatore è stato interrotto in un punto a caso, i LED della porta d'uscita dovrebbero ora visualizzare un altro valore. Il valore mostrato dai LED è il dato qualunque, che era presente sulla porta d'uscita quando è stato premuto il tasto [INTRPT].

H) Ripetete alcune volte il passo G in modo da verificare che il valore presente sui LED è veramente casuale.

I) Premete [INTRPT] e fate così cessare l'esecuzione del programma contatore.

RIASSUNTO

Con questo esperimento si è mostrato l'utilizzo delle interruzioni. Quando è stato premuto INTRTP, si è generata un'interruzione e il microprocessore ha effettuato un salto dal programma contatore alla routine di servizio dell'interruzione, che ha, a sua volta, chiamato la routine beep contenuta in ROM. L'esecuzione della routine beep è continuata per tutto il tempo che è stato premuto il tasto INTRTP. Solo quando tale tasto è stato rilasciato il microprocessore è ritornato al programma contatore.

Il microprocessore contiene diversi registri. Alcuni, come l'accumulatore, sono utilizzati per memorizzare dati, mentre altri, come il program counter (PC contatore di programma) svolgono delle funzioni speciali di controllo. Il PC in particolare indica sempre l'indirizzo dell'istruzione che è stata eseguita.

Il tasto FETCH REG presente nel μLab vi permette di esaminare il contenuto dei diversi registri. In una parte successiva di questo corso vedrete come è possibile utilizzare tale tasto per modificare il contenuto dei diversi registri.

Il tasto FETCH PC riporta sul display il valore precedente del PC e serve per ritornare a un programma dopo che questo è stato formato.

Le subroutine sono programmi che sono utilizzati da altri programmi e vi permettono di suddividere il sistema in diversi piccoli moduli. Per saltare ad una subroutine si usa un'istruzione CALL mentre al termine della subroutine è posta un'istruzione RET. Il flusso del programma, quando viene eseguita un'istruzione RET, ritorna al programma che aveva chiamato la subroutine. Questo modo di operare fa sì che una stessa subroutine possa essere chiamata da più programmi o in punti diversi dello stesso programma.

Le interruzioni sono utilizzate per gestire eventi imprevedibili cui deve essere data una risposta il più possibile veloce. Quando sono abilitate, le interruzioni permettono allo hardware esterno al microprocessore di richiedere che esso esegua immediatamente una certa operazione. Il microprocessore, appena riceve un segnale d'interruzione, interrompe il programma in quel momento in esecuzione e salta ad una routine di servizio della interruzione; solo al termine di questa routine il microprocessore ritorna ad eseguire il programma che era stato interrotto.

DOMANDE

Lezione 5

1. Il principale scopo dell'accumulatore è:
   a. memorizzare temporaneamente dei dati.
   b. indicare quale sarà la prossima istruzione che deve essere eseguita.
   c. selezionare quali interruzioni devono essere abilitate.
   d. memorizzare istruzioni.

2. Il program counter è utilizzato per:
   a. memorizzare dati.
   b. memorizzare istruzioni.
   c. memorizzare l'indirizzo della prossima istruzione da eseguire.
   d. contare i programmi.

3. Il tasto FETCH REG vi permette di esaminare il contenuto dei_____.

4. Per saltare ad una subroutine si usa l'istruzione_____.

5. Al termine di una subroutine si mette l'istruzione_____.

6. Le interruzioni sono utilizzate soprattutto per:
   a. suddividere il programma in più moduli.
   b. dare una risposta veloce a eventi imprevedibili.
   c. rendere più veloce l'esecuzione di un programma.
   d. fermare tutto il sistema.

7. Quando si verifica un'interruzione, il microprocessore:
   a. salta alla routine di servizio dell'interruzione.
   b. si ferma in attesa di un'altra istruzione.
   c. continua ad eseguire il programma principale.
   d. completa il programma in corso e poi si ferma.

LEZIONE 6

All'interno del microprocessore

INTRODUZIONE

Per buona parte di questo corso il microprocessore è stato considerato una *scatola nera:* un dispositivo dotato di certe caratteristiche conosciute, ma la cui struttura interna non sia particolarmente significativa. Tuttavia, per comprendere chiaramente il modo di operare del sistema, è utile capire un poco come funziona interamente il microprocessore. In questa lezione si vuole dare uno sguardo all'interno del microprocessore per vedere in che modo vengono eseguiti i programmi.

ALL'INTERNO DEL MICROPROCESSORE 8085A

Nella Figura 6-1 è riportato uno schema a blocchi semplificato del microprocessore 8085. L'accumulatore è collegato sia al bus dei dati che all'*Unità aritmetico-logica* (ALU, Arithmetic and Logic Unit). Tutte le elaborazioni dei dati, ad esempio incrementare un numero o sommare due valori, sono effettuate dalla ALU.

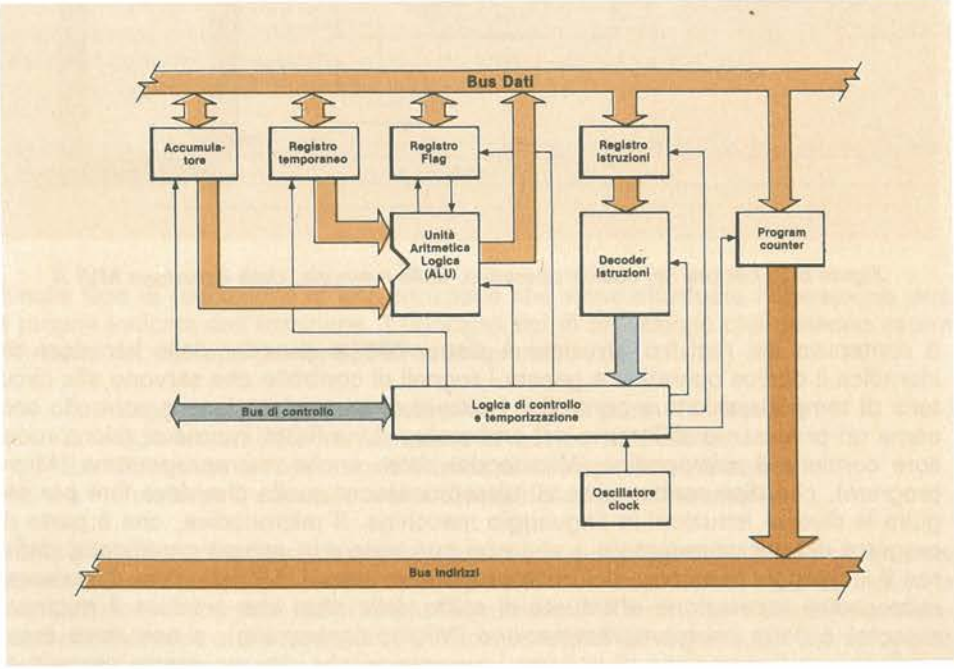


Figura 6-1. Schema a blocchi semplificati dell'8085

L'altro ingresso della ALU è dato dal *Registro temporaneo,* un registro invisibile al programmatore che è controllato automaticamente dalla circuiteria di controllo del microprocessore.

I *flag* (indicatori) sono un gruppo di flip-flop che servono ad indicare certe caratteristiche proprie del risultato dell'ultima operazione effettuata dalla ALU. Così, per esempio; il *flag di zero* è posto a uno se il risultato di una operazione è uguale a zero; allo stato del flag di zero fa riferimento l'istruzione JZ.

Il *Registro istruzioni* (Instruction Register), il *Decodificatore delle istruzioni* (Instruction Decoder), il *Program Counter* (PC, contatore di programma) e la *Logica di controllo e di temporizzazione* (Control and Timing Logic) sono utilizzati per prelevare dalla memoria le istruzioni e per controllare l'esecuzione. Supponete, per esempio, che debba essere letta un'istruzione posta alla locazione 0200. Per prima cosa deve essere letto in memoria il codice operativo: è questa la fase di prelievo (Fetch) di un'istruzione e a tale fase fa riferimento la Figura 6-2. Il contenuto del PC, l'indirizzo 0200, è posto sul bus degli indirizzi in modo da selezionare la locazione di memoria 0200. A questo punto la ROM pone a sua volta il contenuto della locazione 0200 (che dovrebbe essere un codice operativo) sul bus dei dati e il microprocessore immagazzina quindi il codice operativo nel registro istruzioni.

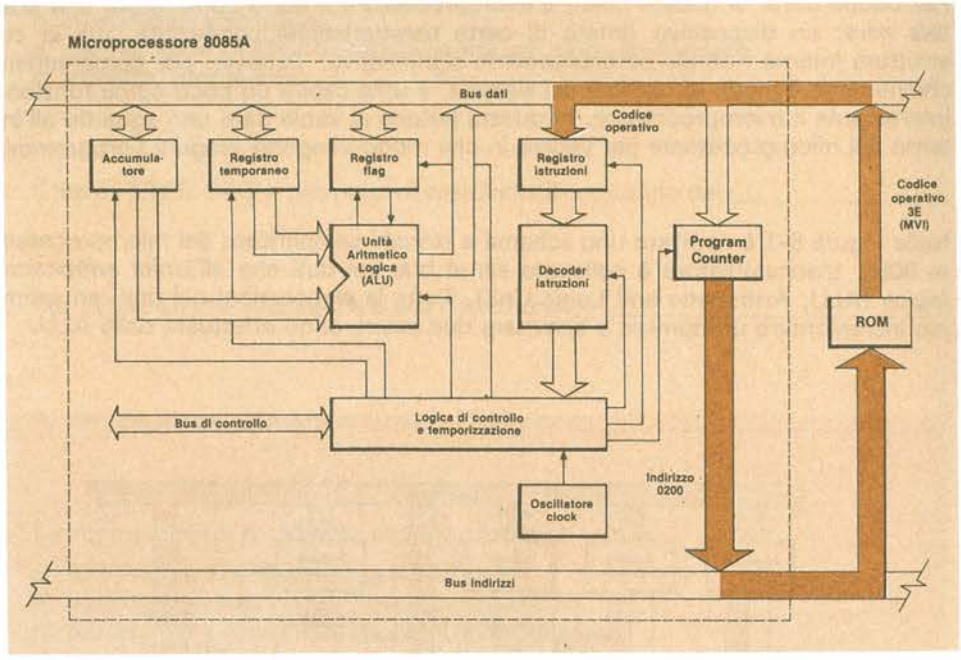


Figura 6-2. Lettura del codice operativo, dalla memoria, della istruzione MVI A.

IL CICLO D'ISTRUZIONE

Il contenuto del registro istruzioni è disponibile al decoder delle istruzioni che identifica il codice operativo e genera i segnali di controllo che servono alla circuiteria di temporizzazione e controllo. I circuiti di temporizzazione e controllo sono come un processore all'interno del processore. Una ROM interna al microprocessore contiene il *microcodice* (Microcode) detto anche *microprogramma* (Microprogram), che dice esattamente al microprocessore quello che deve fare per eseguire le diverse istruzioni in linguaggio macchina. Il microcodice, che è parte del progetto del microprocessore e che non può essere in genere modificato, definisce il linguaggio macchina del microprocessore stesso. L'operazione di scrivere il microcodice (operazione effettuata di solito dalla ditta che produce il microprocessore) è detta *microprogrammazione* (Microprogramming), e non deve essere confusa con l'operazione di scrivere i programmi che devono essere eseguiti dal microprocessore.

Prendiamo per esempio l'istruzione MVI A. Per prima cosa la logica di controllo e temporizzazione riconosce il codice operativo 3E e decide quindi che tale istruzione prevede anche un byte di dato, per cui incrementa l'indirizzo presente nel PC. Viene così letto e trasferito nell'accumulatore (Figura 6-3) il contenuto della locazione di memoria indicata dal PC (il secondo byte dell'istruzione, che contiene il dato che deve essere caricato nell'accumulatore).

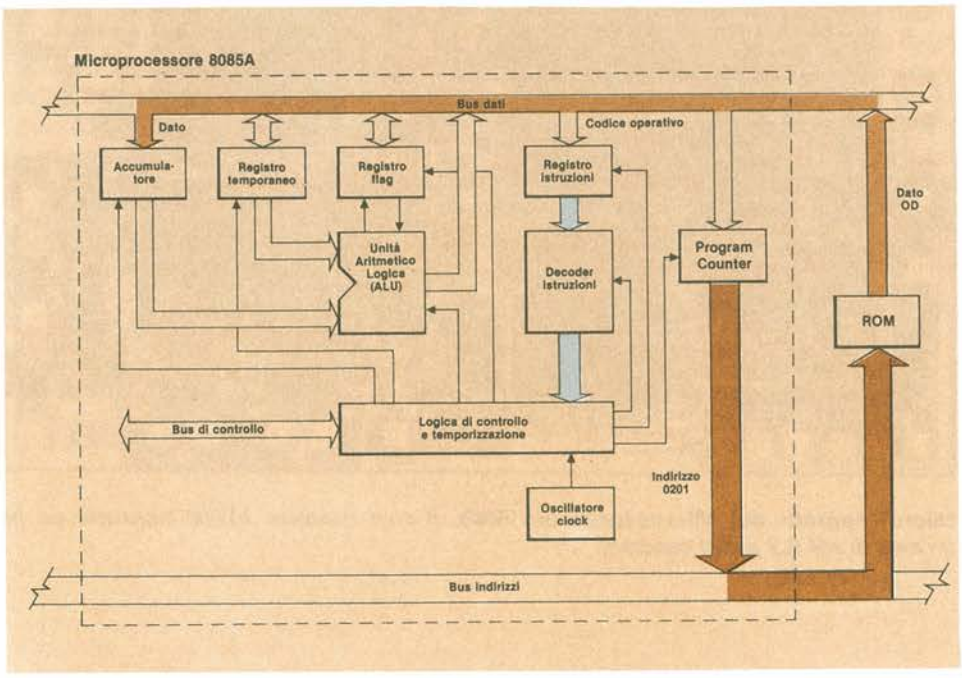


Figura 6-3. Lettura del dato dell'istruzione MVI A

Il microprogramma indica ora alla logica di controllo che l'istruzione è stata completata. Viene incrementato il PC e viene letto e trasferito nel registro istruzioni il prossimo byte del programma (cioè il prossimo codice operativo).
Inizia così l'esecuzione di questa nuova istruzione.

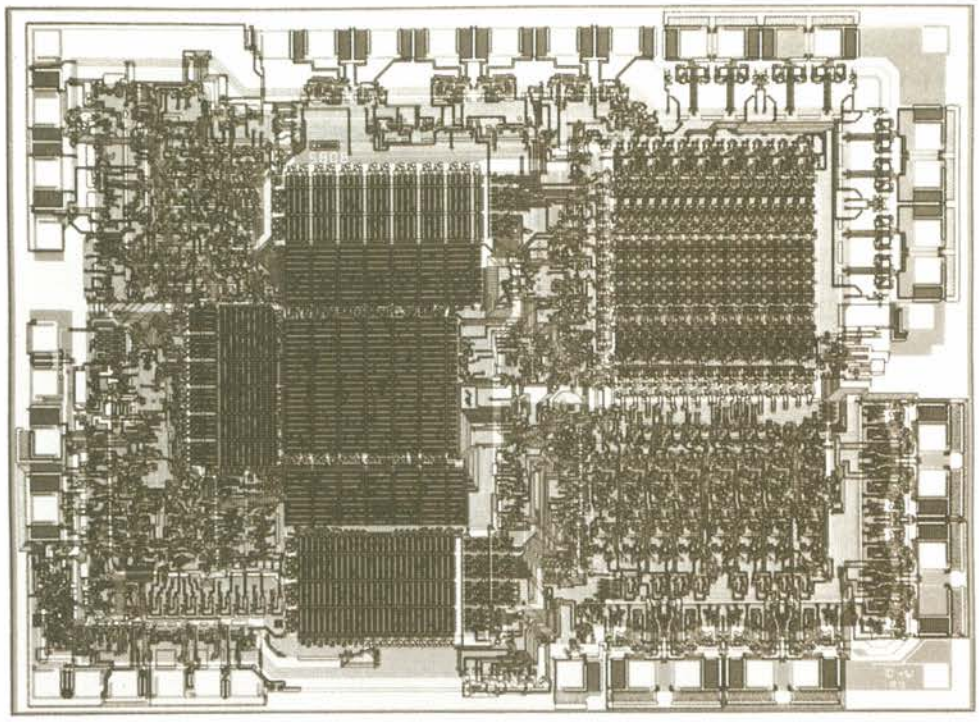
Questa sequenza di operazioni, che il microprocessore effettua in continuazione, è detta *ciclo di prelievo-esecuzione* (Fetch-Execute Cycle).

ESECUZIONE DELLE ISTRUZIONI

È nella fase di esecuzione di una istruzione che viene effettuata l'operazione vera e propria indicata dall'istruzione. I principali tipi di operazione che possono essere effettuati dal microprocessore 8085 sono quattro:

1. Leggere un dato dalla memoria o da una porta d'ingresso.
2. Scrivere un dato in memoria o in una porta d'uscita.
3. Effettuare un'operazione interna al microprocessore stesso.
4. Trasferire il controllo a una diversa locazione di memoria.

Non c'è molto da dire riguardo ai primi due tipi, la cui spiegazione è già chiaramente contenuta nelle definizioni date. Il terzo tipo (effettuare operazioni interne) implica la manipolazione dei registri (ad esempio l'accumulatore) senza che siano effettuati accessi alla memoria o alle porte di I / O. Il contenuto di un registro, ad esempio, può essere trasferito ad un altro registro, oppure essere incrementato o decrementato. Il quarto gruppo comprende, infine, istruzioni come JMP, CALL e RET.



Microfotografia del Microprocessore 8085. *Il chip contiene 20.000 transistor ed ha un'area di soli 0,2 pollici quadrati.*

CICLI MACCHINA

L'operazione di prelievo e quella di esecuzione delle istruzioni sono divise in *cicli macchina.* Il primo ciclo macchina di ogni istruzione è sempre il prelievo del codice operativo. Un ulteriore ciclo macchina è poi necessario per ogni accesso in memoria o a porte di I / O, in modo da permettere il trasferimento dei dati. In un ciclo macchina, viene dapprima posto l'indirizzo sul bus degli indirizzi, successivamente avviene il trasferimento dell'informazione sul bus dei dati. La maggior parte delle operazioni interne al microprocessore (è il caso ad esempio di una operazione di incremento dell'accumulatore) sono portate a termine nello stesso ciclo macchina di prelievo del codice operativo. Così un'istruzione semplice, come INR A, richiede solo un ciclo macchina, mentre l'istruzione STA ha bisogno di quattro cicli macchina: tre per leggere l'istruzione stessa e uno per trasferire l'accumulatore in memoria.

Ogni volta che viene premuto il tasto hardware step, presente nel μLab, viene eseguito un solo ciclo macchina. Un ulteriore passo (dopo che sono stati letti tutti i byte di una istruzione) è necessario per l'esecuzione di istruzioni che effettuano un accesso alla memoria o alle porte di I / O. Questo ulteriore passo è necessario perché il microprocessore possa definire lo stato del bus degli indirizzi e possa trasferire il dato sul bus dei dati. Le operazioni interne, invece, che sono eseguite nella stessa fase di prelievo dell'istruzione, non hanno bisogno di questo ulteriore passo per effettuare la fase di esecuzione.

ESECUZIONE DI UN PROGRAMMA

Di solito il microprocessore procede leggendo sequenzialmente la memoria, una locazione dopo l'altra, ed eseguendo le operazioni indicate dalle diverse istruzioni. Un comportamento diverso si verifica quando vengono eseguite istruzioni di salto (JUMP), di chiamata (CALL) o di ritorno. Un'altra eccezione si ha quando si verifica un'interruzione. In tutti questi casi il microprocessore interrompe il flusso sequenziale e incomincia ad eseguire istruzioni prelevandole da un indirizzo diverso.

Notate che nella memoria i codici operativi e i dati sono variamente mescolati. Ad un certo indirizzo può essere presente un codice operativo, all'indirizzo successivo un indirizzo di salto, a quello successivo ancora un codice operativo e a quello seguente una parte di un dato. È compito del programmatore assicurarsi che la sequenza di codici operativi e di dati presente in memoria sia corretta. Il microprocessore è in grado di distinguere solo in base al contesto in cui li trova. Ricordate che i codici operativi, indirizzi di salto e dati sono tutti delle configurazioni di bit poste in memoria. Ancora, ogni informazione posta in memoria è letta esattamente allo stesso modo ed i trasferimenti delle informazioni avvengono sempre tramite lo stesso bus dei dati. Il microprocessore deve perciò ad ogni istante sapere se sta leggendo un dato o un codice operativo per poterli così trattare in modo corretto. Il microprocessore assume che la prima locazione che legge contenga un codice operativo e da questo punto in poi procede. Se il codice operativo prevede un byte di dato, il microprocessore «sa» (grazie al microcodice interno) che il byte successivo sarà un dato e lo tratterà di conseguenza. Il byte dopo il dato viene invece considerato il nuovo codice operativo. Se un dato è erroneamente interpretato come codice operativo, il sistema *perderà con buona probabilità il controllo.*

ESPERIMENTO 6-1

Come opera il bus

INTRODUZIONE

Viene inserito nel μLab il programma, utilizzato nell' Esperimento 4-3, che legge i dati dalla porta d'ingresso e li scrive sulla porta d'uscita; servendosi del modo hardware step, tale programma viene poi eseguito passo passo. Per ogni istruzione potrete così seguire il ciclo prelievo (fetch)-esecuzione (execute) e, servendovi dei LED posti sul bus degli indirizzi, sul bus dei dati e sulle linee di stato, potrete così vedere le operazioni che si svolgono sui diversi bus. Ad ogni passo dovrete inserire in una tabella le informazioni lette sui LED e, al termine, questa tabella vi mostrerà i diversi passi dell'esecuzione di un programma.

PROCEDIMENTO

A) Introducete da tastiera il seguente programma:

Indirizzo	Contenuto	Label	Istruzione	Commenti
0900	3A	START:	LDA 2000	;Leggi la porta d'ingresso.
0901	00			
0902	20			
0903	32		STA 3000	;Trasferisci il dato sulla porta di uscita.
0904	00			
0905	30			
0906	C3		JMP START	;Salta all'inizio.
0907	00			
0908	09			

B) Verificate che il programma sia stato correttamente memorizzato.

C) Premete [FETCH ADRS] [0] [9] [0] [0] [HDWR STEP]. Con questa operazione il μLab si trova ad operare nel modo hardware step, partendo dall'indirizzo d'inizio del programma.

D) I LED degli indirizzi visualizzano ora il primo indirizzo del programma (0900) mentre i LED dei dati visualizzano il primo codice operativo (3A). I LED di stato indicano che è in corso un'operazione di lettura (READ) in RAM. Notate che la prima linea della Tabella 6-1 riporta esattamente le informazioni visualizzate sui tre gruppi di LED e che la linea successiva ripete le stesse informazioni convertite in esadecimale.

E) Premete [HDWR STEP]. Viene letto dalla RAM il secondo byte dell'istruzione LDA (i bit di ordine basso dell'indirizzo della porta, 00). Inserite nella terza linea della Tabella 6-1 le informazioni che leggete sui tre gruppi di LED. Convertite poi l'indirizzo e il dato in esadecimale ed inserite nella riga successiva i valori esadecimali così ottenuti.

F) Premete [HDWR STEP]. Viene così letto l'ultimo byte della istruzione LDA (i bit di ordine alto dell'indirizzo della porta 20). Inserite nella Tabella 6-1 le informazioni ricavate dai LED.

	Indirizzo																Dato								Read	Write	ROM	RAM	In Port	Out Port
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0						
Binario	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	1	0	0
Hex	0				9				0				0				3				A				Lettura della RAM					
Binario																														
Hex																														
Binario																														
Hex																														
Binario																														
Hex																														
Binario																														
Hex																														
Binario																														
Hex																														
Binario																														
Hex																														
Binario																														
Hex																														
Binario																														
Hex																														
Binario																														
Hex																														

Tabella 6-1. Tabella del programma dell'Esperimento 6-1

ESPERIMENTO 6-1

(continuazione)

G) Premete [HDWR STEP]. È questa la fase di esecuzione della istruzione LDA. I LED degli indirizzi visualizzano ora l'indirizzo della porta, 2000. Spostate a piacere gli interruttori della porta d'ingresso. Poiché è in corso la lettura di tale porta, il dato presente sugli interruttori è visualizzato pari pari sui LED del bus dei dati. Notate che i LED di stato indicano che è in corso una lettura (READ) dalla porta d'ingresso (INPUT). Introducete tali informazioni nella Tabella 6-1.

H) Premete [HDWR STEP]. Riportate l'indirizzo, il dato e lo stato attuale nella Tabella 6-1.

I) Ripetete il passo H finché il programma non ritorna all'inizio (indirizzo 0900). Seguite attentamente il modo di operare del programma, osservando i LED binari, e prestate particolare attenzione alla fase di esecuzione dell'istruzione STA.

J) Se non vi fidate delle informazioni che avete riportato in tabella, potete ora ripetere l'intero procedimento. Studiate le informazioni contenute nella Tabella 6-1, in modo da identificare la sequenza prelievo-esecuzione per ogni istruzione del programma. Quale istruzione non ha bisogno, per essere eseguita, di un ulteriore ciclo?

K) Premete [RESET] [FETCH ADRS] [0] [9] [0] [0].

L) Premete ora più volte [INSTR STEP] in modo da eseguire, istruzione dopo istruzione, tutto il programma. Notate che questa volta un solo passo è sufficiente perché un'istruzione sia letta (cioè tutti i byte che la compongono) ed eseguita. In questo modo perdono di significato i LED binari degli indirizzi, dei dati e di stato. Confrontate la sequenza degli indirizzi visualizzata sul display con quella che avete riportato nella Tabella 6-1.

RIASSUNTO

Servendosi del modo hardware step e osservando i LED degli indirizzi, dei dati e di stato, avete potuto seguire nei dettagli come viene eseguito un programma. Dovreste ora essere in grado di confrontare i dati che avete riportato nella Tabella 6-1 con il listing del programma presentato nel passo A di questo esercizio. Ogni istruzione richiede un ciclo macchina per ogni byte che l'istruzione stessa contiene. Le istruzioni STA e LDA hanno bisogno di un ulteriore ciclo macchina di esecuzione. Le altre istruzioni (l'istruzione di JMP, per esempio) sono invece eseguite in numero di cicli pari a quello di lettura dell'istruzione stessa. Il modo hardware step vi permette di vedere singolarmente ciascuno di questi cicli. Il modo instruction step invece riassume tutti i cicli macchina di una istruzione in un passo solo. Notate inoltre che il bus dei dati è utilizzato per trasferire i dati in entrambe le direzioni, e che il bus degli indirizzi è utilizzato per indirizzare sia la memoria che le porte di I / O.

Lezione 6

I programmi sono immagazzinati in memoria come sequenze di numeri binari. Il microprocessore esegue un programma leggendo la prima locazione di memoria e interpretandola come codice operativo. Se il codice operativo implica che il byte successivo (o i due successivi) sia un dato (o siano un indirizzo), il microprocessore legge tale informazione dalla locazione di memoria corrispondente. Una volta che tutti i byte dell'istruzione sono stati prelevati, il microprocessore può eseguire l'istruzione. È allora letto ed interpretato come codice operativo il byte presente nella locazione di memoria successiva e l'operazione continua. Si è così completato il ciclo d'istruzione, detto anche ciclo di prelievo-esecuzione. Nel modo instruction step del μLab viene eseguito un ciclo d'istruzione completo per volta, per cui tale modo è utilizzabile per seguire il flusso complessivo del programma. Il modo hardware step, che visualizza i singoli cicli macchina, è invece utilizzato per osservare i dettagli di comportamento di ciascuna istruzione.

Con il microprocessore 8085 possono essere effettuate operazioni che trasferiscono dati tra i registri del microprocessore e la memoria o le porte di I / O, o operazioni che alternano il contenuto dei registri del microprocessore stesso. In questo modo i dati sono in genere acquisiti dal microprocessore attraverso un'operazione di lettura dalla memoria o da una porta di ingresso, sono elaborati interamente al microprocessore e sono poi trasferiti ancora alla memoria o a una porta d'uscita.

DOMANDE

Lezione 6

1. L'operazione di lettura di un codice operativo dalla memoria è detta operazione di _____.

2. Il microprocessore sa quali byte deve interpretare come codici operativi perché:
   a. ogni byte è un codice operativo.
   b. il terzo byte è sempre un codice operativo.
   c. in ogni codice operativo è indicato il numero dei byte di informazione successivi.
   d. il programma deve specificare quali byte sono dei codici operativi.

3. Per osservare i dettagli di comportamento delle singole istruzioni si deve usare il modo_____ step.

4. Quante volte dovete premere il tasto hardware step (ovvero quanti cicli macchina sono necessari) per eseguire una istruzione?
   a. il numero di byte che formano un'istruzione.
   b. il numero di byte che formano un'istruzione più tre.
   c. il numero di byte che formano un'istruzione più il numero di riferimento in memoria o a porte di I / O necessari per eseguire l'istruzione stessa.
   d. sempre 3.

5. La funzione dell'ALU è:
   a. interpretare i codici operativi.
   b. effettuare le operazioni logiche e aritmetiche.
   c. controllare il bus degli indirizzi.
   d. calcolare il numero di cicli macchina richiesti.

III
HARDWARE DEL SISTEMA A MICROPROCESSORE

Questa sezione tratta in dettaglio l'hardware del microprocessore. La prima lezione introduce i circuiti base del microprocessore e se ne raccomanda la lettura a tutti gli studenti. Le lezioni successive si riferiscono ai concetti hardware della decodifica degli indirizzi, memorie, periferiche, controllo di sistema, e considerazioni dal punto di vista elettrico. Pur trattando come base della discussione il progetto del μLab, vengono descritti altri progetti alternativi.

LEZIONE 7

Concetti fondamentali di hardware

INTRODUZIONE

In questa lezione sono descritti gli elementi hardware fondamentali presenti in un sistema a microprocessore. Sono discusse la struttura a bus ed il problema della decodifica degli indirizzi; invece che descrivere alcuni progetti specifici si insiste soprattutto sulla spiegazione dei blocchi fondamentali che costituiscono un sistema tipico. I circuiti di un «vero» sistema a microprocessore saranno più dettagliatamente descritti nelle prossime lezioni.

IL CONCETTO DI BUS

I sistemi a microprocessore sono progettati intorno ai bus, strutture che non si trovano di solito nei progetti tradizionali a logica cablata. In un sistema a microprocessore ci sono parecchi dispositivi che devono scambiare dati con il processore. La Figura 7-1 mostra come questo problema potrebbe essere risolto con tecniche di progetto tradizionali. Il processore ha tante uscite di dati quanti sono i dispositivi presenti, mentre un multiplex permette di selezionare un certo dispositivo come ingresso. Questa soluzione diventa sempre più complessa ed ingestibile al crescere del numero dei dispositivi presenti. I dati vengono di solito trasferiti otto bit alla volta, per cui ad ogni percorso devono essere associate otto linee.

Così, per il semplice sistema costituito da tre dispositivi riportato in figura, sono già necessarie 48 linee: 24 per l'ingresso dei dati e 24 per l'uscita dei dati. Un sistema più complesso può essere costituito da dozzine di dispositivi di memoria e di porte di I/O e può quindi richiedere centinaia di linee di collegamento.

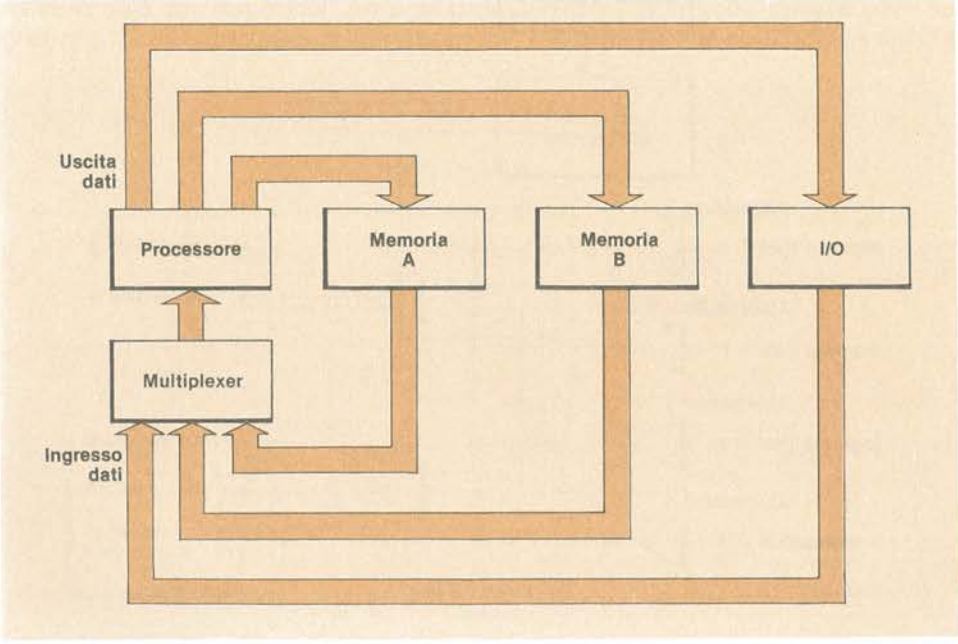


Figura 7-1. Trasferimenti di dati secondo tecniche di progetto tradizionali

È possibile risolvere altrimenti questo problema utilizzando un bus, soluzione questa presentata in Figura 7-2. Si noti come si semplificano i collegamenti. Per collegare tutti i dispositivi è utilizzato un solo insieme di otto linee (un bus appunto) attraverso cui i dati possono viaggiare sia verso il processore che dal processore verso l'esterno. È possibile espandere all'infinito questa struttura, aumentando solo di poco la complessità dei collegamenti. Come conseguenza diretta di questa tecnica, dato che tutti i dispositivi hanno le stesse linee di dati, è necessario che uno solo tra di essi possa, ad un certo istante, presentare i dati. Sono le linee di controllo e di indirizzo (pilotate dal microprocessore) che controllano che venga selezionato un solo dispositivo per volta.

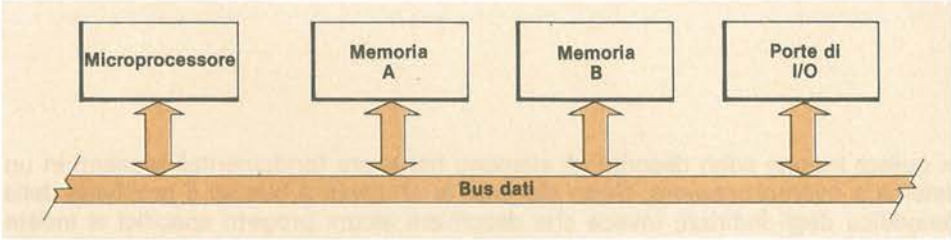


Figura 7-2. Trasferimento dati attraverso un bus in modo da ridurre il numero dei collegamenti

IL BUS A TRE STATI

È possibile che il bus dei dati sia condiviso da più dispositivi se si utilizzano dei *driver* (dispositivi cioè per pilotare linee di segnale) *a tre stati*. Per semplicità, discuteremo dapprima il problema di un bus formato da una sola linea; i concetti di base rimangono comunque gli stessi, qualunque sia il numero di linee che compongono il bus. (Un bus dei dati ha tipicamente otto linee).

Il *bus a tre stati* è come una linea telefonica: al bus possono essere collegati dispositivi che vogliono «parlare» (che chiameremo d'ora innanzi parlatori) e dispositivi che vogliono ascoltare (che chiameremo d'ora innanzi ascoltatori). Nella Figura 7-3 è mostrato il bus di un circuito digitale, in cui sono presenti quattro parlatori (i driver a tre stati) e due ascoltatori (i gate soliti).

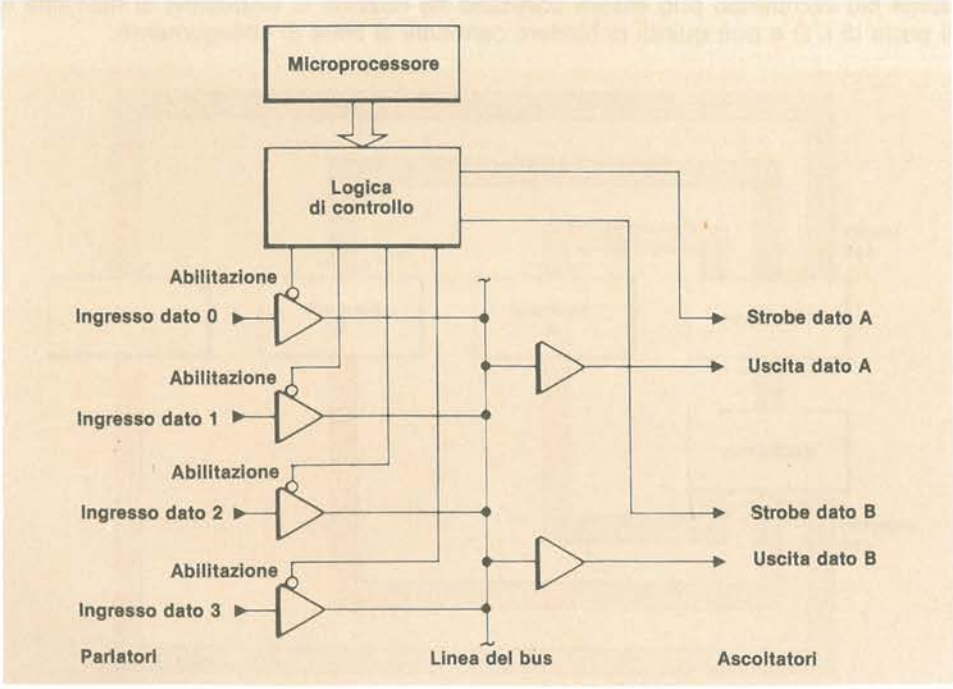


Figura 7-3. Bus a tre stati, formato da una sola linea, con quattro dispositivi «parlatori» e due «ascoltatori»

La logica di controllo fa sì che sia attivo solo un driver (parlatore) per volta. Se più di un parlatore fosse contemporaneamente abilitato, i dati sul bus non avrebbero significato. Quando abilitato, un driver pone sul bus il dato che è presente al suo ingresso; tutti gli altri driver sono invece disabilitati e le loro uscite sono poste in uno stato ad alta impedenza (floating, cioè fluttuanti, aperte) in modo da non alterare lo stato logico del bus. (Per una spiegazione della simbologia usata in questo corso, fate riferimento all'Appendice D)

Sul bus sono presenti diversi ascoltatori. Poiché il loro compito è solo quello di ascoltare, più di un ascoltatore può essere, allo stesso istante, abilitato. In generale, tuttavia, il dato presente sul bus sarà indirizzato ad un solo ascoltatore. Sono i segnali generati dalla logica di controllo (gli strobe dei dati, impulsi cioè di campionamento dei dati) ad indicare, agli ascoltatori che si vogliono selezionare, che il dato presente sul bus è diretto a loro. Uno strobe dei dati può essere ad esempio utilizzato per memorizzare in un flip-flop il dato presente sul bus. Gli ingressi della logica di controllo saranno il bus degli indirizzi ed il bus di controllo provenienti dal microprocessore. (Figura 7-4).

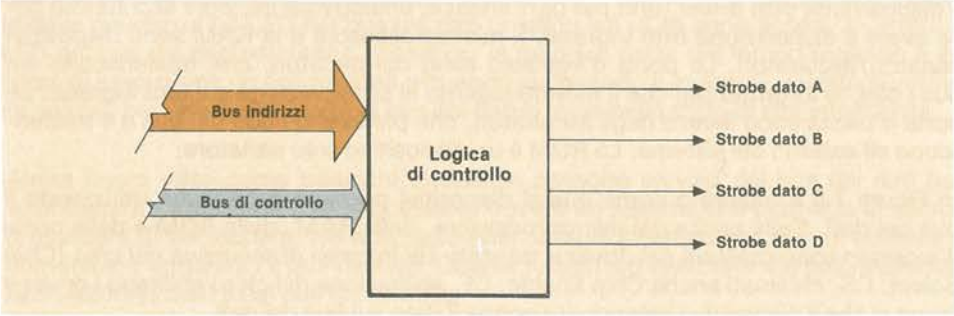


Figura 7-4. Logica di controllo che seleziona i dispositivi che sono interessati al trasferimento dei dati

I dispositivi finora descritti sono di tipo *unidirezionale*, cioè sono o parlatori od ascoltatori, ma non le due cose insieme. Esistono anche dispositivi *bidirezionali* che possono essere sia parlatori che ascoltatori. La Figura 7-5 mostra appunto un bus con due parlatori/ascoltatori. Se ne potrebbero avere parecchi, ma per semplicità sono mostrati solo due dispositivi. Ad ogni parlatore/ascoltatore sono associati due segnali di controllo: il segnale di abilitazione dell'uscita, che serve ad abilitare il driver a

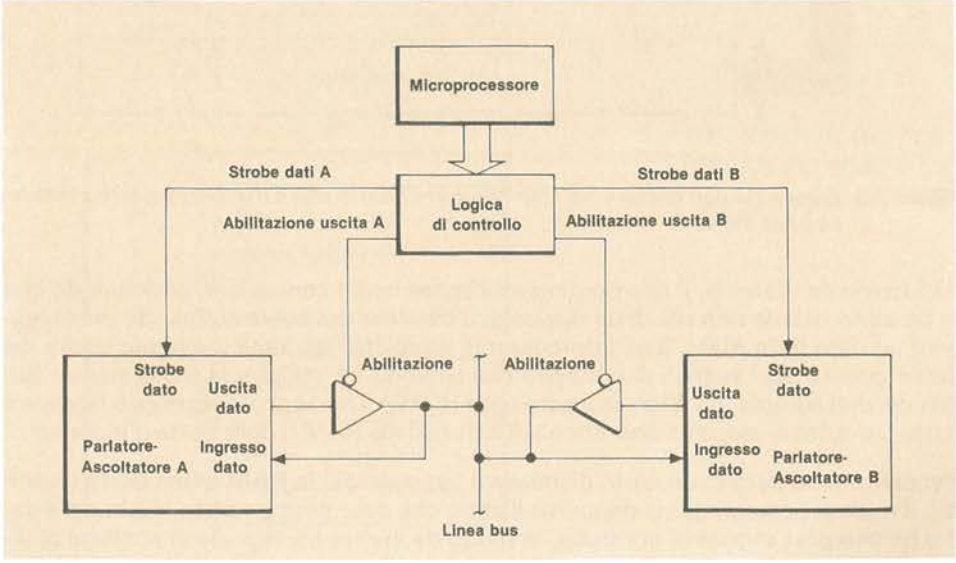


Figura 7-5. Parlatori-ascoltatori bidirezionali collegati alla linea del bus

tre stati, e un segnale di strobe dei dati per controllare l'ingresso. Una RAM è un esempio di dispositivo bidirezionale in cui i dati possono essere letti o scritti.

Per vedere come opera un processore, si faccia riferimento alla Figura 7-5, in cui il dispositivo A deve inviare dei dati al dispositivo B.

La logica di controllo fa sì che la linea di abilitazione dell'uscita A diventi vera e la linea di abilitazione dell'uscita B falsa. Successivamente, trascorso un certo tempo di assestamento (*Settling Time*) perché i dati possano raggiungere l'ingresso dei dati del dispositivo B, la logica di controllo genera un impulso sulla linea di strobe dei dati B. Il dispositivo B legge così dal bus i dati presentati dal dispositivo A. Notate che al bus possono essere collegati parecchi altri dispositivi che rimarranno però inefficaci finché le loro linee di abilitazione saranno false.

IL BUS DEI DATI

Il bus dei dati di un sistema a microprocessore è un bus bidirezionale a tre stati. Tale bus è praticamente identico al bus formato da una sola linea del nostro esempio precedente, unica differenza il fatto di essere costituito da otto linee anziché da una linea sola. Ogni parlatore, per poter utilizzare tutte le linee del bus dei dati, deve avere a disposizione otto driver (uno per ogni linea) e, analogamente, ogni ascoltatore deve avere a disposizione otto ingressi. Il microprocessore e la RAM sono dispositivi parlatori/ascoltatori. Le porte d'ingresso sono dei parlatori, che trasferiscono sul bus i dati, le informazioni che il mondo esterno al sistema porta sui loro ingressi. Le porte d'uscita sono invece degli ascoltatori, che prelevano i dati dal bus e li trasferiscono all'esterno del sistema. La ROM è un dispositivo solo parlatore.

In Figura 7-6 è mostrato come questi dispositivi possano colloquiare utilizzando il bus dei dati. Sulle uscite del microprocessore, della RAM, della ROM e delle porte d'ingresso sono presenti dei driver a tre stati. Gli ingressi di *selezione del chip* (Chip Select, CS, chiamati anche Chip Enable, CE, abilitazione del chip) abilitano i driver e fanno sì che il dispositivo selezionato ponga il dato sul bus dei dati.

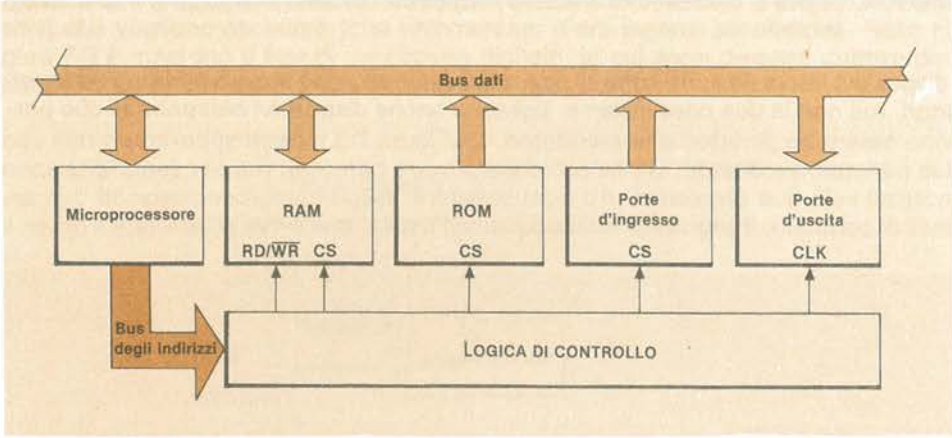


Figura 7-6. Dispositivi con uscita a tre stati che comunicano con il microprocessore attraverso il bus dei dati

All'interno del sistema, il microprocessore opera come controllore, assicurando che in un certo istante non più di un dispositivo cerchi di accedere al bus. Se vuole leggere un dato dalla ROM, il microprocessore disabilita dapprima le proprie uscite dei dati e genera poi i segnali di controllo che servono ad abilitare la ROM stessa. Sul bus dei dati compaiono allora le uscite della ROM e il microprocessore può leggere il dato. Del tutto analoga è l'operazione di lettura della RAM o della porta d'ingresso.

Per scrivere un dato in un certo dispositivo (ad esempio la RAM o una porta d'uscita), il microprocessore pone dapprima il dato, che deve essere scritto, sul bus dei dati e genera poi i segnali di controllo, in modo da inviare un segnale di scrittura al dispositivo scelto. Con l'impulso di *scrittura* il dato viene memorizzato (Latched) internamente al dispositivo.

In generale i dati passano attraverso il microprocessore. Così per trasferire un dato dalla porta d'ingresso alla RAM, il microprocessore legge in un primo momento il dato sulla porta d'ingresso e lo scrive successivamente nella RAM. I dati, poichè non possono essere direttamente trasferiti dalla porta d'ingresso alla RAM, devono essere temporaneamente immagazzinati all'interno del microprocessore.

Riassumendo, in un sistema a microprocessore il bus dei dati è utilizzato in tutti i trasferimenti di dati che si verificano. Tutti i dispositivi utilizzano lo stesso bus. La logica di controllo, che opera in base ai segnali generati dal microprocessore, comanda i singoli dispositivi, informando ad esempio ciascuno di essi quando è il momento opportuno per porre un dato sul bus o per leggere un dato dal bus.

Nel μLab il bus dei dati è formato da otto linee. Il microprocessore 8085 (il microprocessore presente nel μLab) può gestire otto bit di dato per volta e si dice, perciò, che è un processore a otto bit. Esistono altri tipi di microprocessori che trattano dati di lunghezza maggiore o minore. Dei primi microprocessori realizzati, parecchi avevano un bus dei dati di quattro bit mentre alcuni tra i più recenti hanno un bus dei dati di 16 bit.

Quando nessun dispositivo è abilitato, il bus dei dati è aperto e richiamato alto tramite delle resistenze e i LED del bus dei dati presenti sul μLab sono accesi. In effetti i LED del bus dei dati sembrano accesi per la maggior parte del tempo (quando è in corso di esecuzione un programma o quando sta girando il monitor): questo si verifica poiché, per una buona parte, il bus dei dati si trova nello stato di alta impedenza.

IL BUS DEGLI INDIRIZZI

Avete finora visto come parecchi dispositivi possono servirsi del bus dei dati per scambiarsi informazioni. È ora necessario trovare un sistema che permetta al microprocessore di selezionare, tra tutti, solo quel certo dispositivo che si vuol abilitare a comunicare attraverso il bus dei dati. Questa funzione è realizzata dal *bus degli indirizzi* (Address bus) e dal bus di controllo.

Essendo unidirezionale, il bus degli indirizzi opera in modo più semplice del bus dei dati. Ad ogni locazione, e ad ogni porta di I/O, è associato in modo univoco un indirizzo. Il microprocessore, prima che si possa verificare il trasferimento di un qualunque dato (attraverso il bus dei dati), deve presentare all'esterno un indirizzo. L'indirizzo specifica proprio la locazione di memoria (o la porta di I/O) cui il processore intende accedere. In questo modo il processore può selezionare qualunque parte del sistema con la quale intenda comunicare.

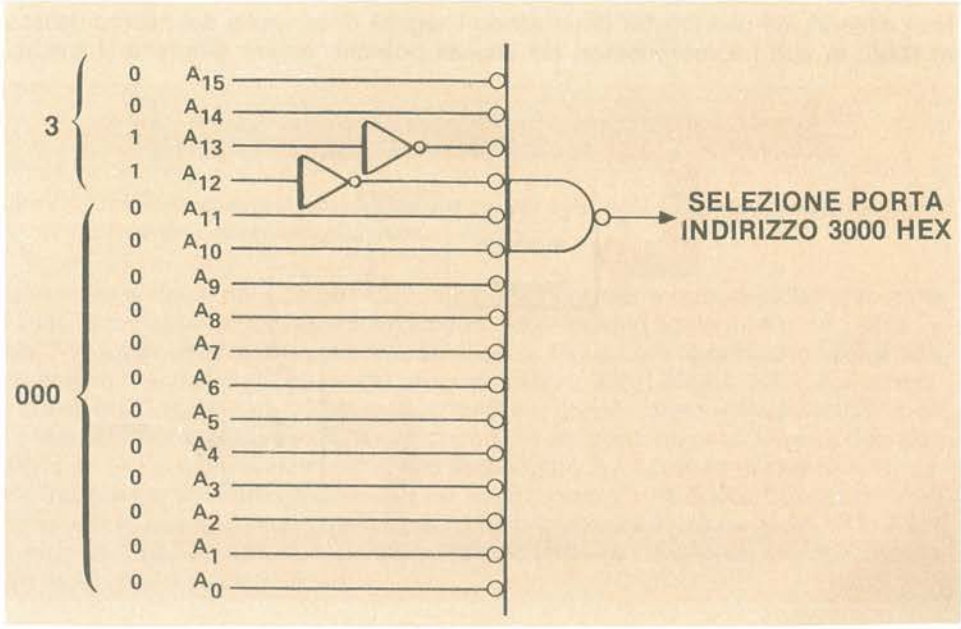


Figura 7-7. Decoder degli indirizzi che controlla la porta di indirizzo 3000

Il bus degli indirizzi del microprocessore 8085 ha sedici linee, e permette così di indirizzare $2^{16} = 65.536$ locazioni di memoria o porte di I/O. Tali linee sono denominate A0, A1, A2,... A15, dove A0 indica il bit meno significativo.

DECODER DEGLI INDIRIZZI

Il *decoder degli indirizzi* (Address Decoder, decodificatore), è una parte della logica di controllo e genera dei segnali di selezione dispositivo quando sul bus degli indirizzi è presente un certo indirizzo (o un certo insieme di indirizzi). Ad esempio, in Figura 7-7 è mostrato un decoder di indirizzo che seleziona l'indirizzo 3000 Hex (0011 0000 0000 0000 in binario).
L'uscita di questo decoder è vera (stato logico 0) solo quando sul bus degli indirizzi è presente tale valore. Questa uscita può così essere utilizzata per abilitare una porta cui sia stato assegnato l'indirizzo 3000.

IL BUS DI CONTROLLO

Avete visto come con il bus degli indirizzi si possa selezionare una ben precisa locazione di memoria o porta di I/O e come il bus dei dati serva a trasferire i dati. Tutta l'operazione è coordinata dal bus di controllo, che è costituito da un certo numero di segnali, la maggior parte dei quali sono generati dal microprocessore (solo pochi sono segnali di ingresso al microprocessore). In questa lezione verranno discussi solo i segnali che controllano le operazioni di lettura e di scrittura nelle porte di I/O e in memoria. Gli altri segnali di controllo, che servono per gestire interruzioni, memorie lente ed operazioni di accesso diretto in memoria (Direct Memory Access, DMA), saranno invece descritti nella Lezione 10.

I due principali segnali di controllo, che vengono generati dal microprocessore 8085, sono quelli di $\overline{\text{READ}}$ (Lettura) e di $\overline{\text{WRITE}}$ (scrittura). Se il segnale di $\overline{\text{READ}}$ è basso, vuol dire che è in corso un'operazione di lettura: il microprocessore segnala così che il dispositivo indirizzato deve porre un dato sul bus dei dati. Se è invece il segnale di $\overline{\text{WRITE}}$ ad essere basso, vuol dire allora che è in corso un'operazione di scrittura: in questo caso è il microprocessore stesso che pone un dato sul bus dei dati e segnala quindi al dispositivo interessato di memorizzare tale dato. I LED di stato READ e WRITE, presenti sul μLab, sono collegati a questi segnali e sono accesi quando il corrispondente segnale è vero (basso).

La differenza più significativa tra il bus di controllo e i bus dei dati e degli indirizzi è che nel bus di controllo ogni linea è specializzata ad un'unica funzione, diversa da quella delle altre linee. Nel caso dei bus degli indirizzi e dei dati, invece, ogni linea porta lo stesso tipo di informazione (un bit dell'indirizzo o un bit del dato).

Non dimenticate che stiamo descrivendo i segnali di controllo del microprocessore 8085: in altri microprocessori tali segnali possono essere differenti. I trasferi-

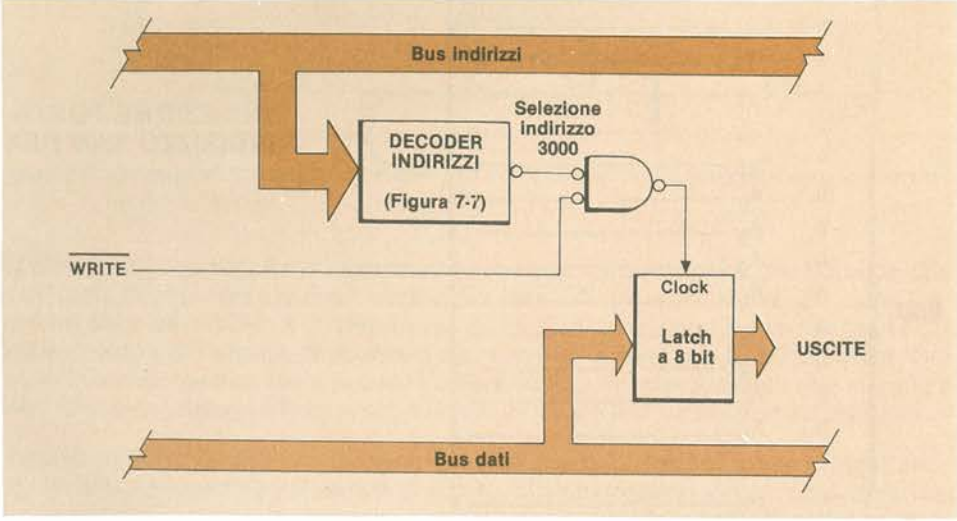


Figura 7-8. Il dato presente nel bus dei dati è immagazzinato in un latch, ogni volta che il microprocessore scrive nell'indirizzo 3000.

menti dati saranno identici, ma potranno essere effettuati secondo modalità diverse. Fate riferimento alla Lezione 20 per la descrizione dei segnali di controllo di altri microprocessori.

PORTE D'USCITA

In Figura 7-8 è mostrata una porta d'uscita a latch (con tale nome, intraducibile, si indica in inglese un particolare tipo di flip-flop) cui è assegnato l'indirizzo 3000. Quando sul bus degli indirizzi è presente il valore 3000 (riconosciuto dal decoder degli indirizzi) e quando il segnale $\overline{\text{WRITE}}$ di controllo ha una transizione basso-alto, viene generato un impulso che va ad un ingresso (detto ingresso di clock) del latch. Grazie a tale impulso il dato presente sul bus dei dati è così memorizzato nel latch stesso. Scrivendo il dato all'indirizzo 3000, il microprocessore può perciò far sì che un certo dato, definito dal programma, sia presentato sulle uscire del latch.

PORTE D'INGRESSO

Le porte d'ingresso sono collegate in modo del tutto analogo (Figura 7-9). Per far sì che la porta sia abilitata, l'uscita del decoder degli indirizzi è questa volta posta in AND con il segnale di $\overline{\text{READ}}$ invece che con quello di $\overline{\text{WRITE}}$. La porta d'ingresso è costituita da un driver a tre stati di otto linee che, quando è abilitato, pone i segnali d'ingresso sul bus dei dati. Effettuando una operazione di lettura all'indirizzo opportuno il microprocessore può leggere così i segnali d'ingresso e immagazzinare tale dato in uno dei suoi registri interni.

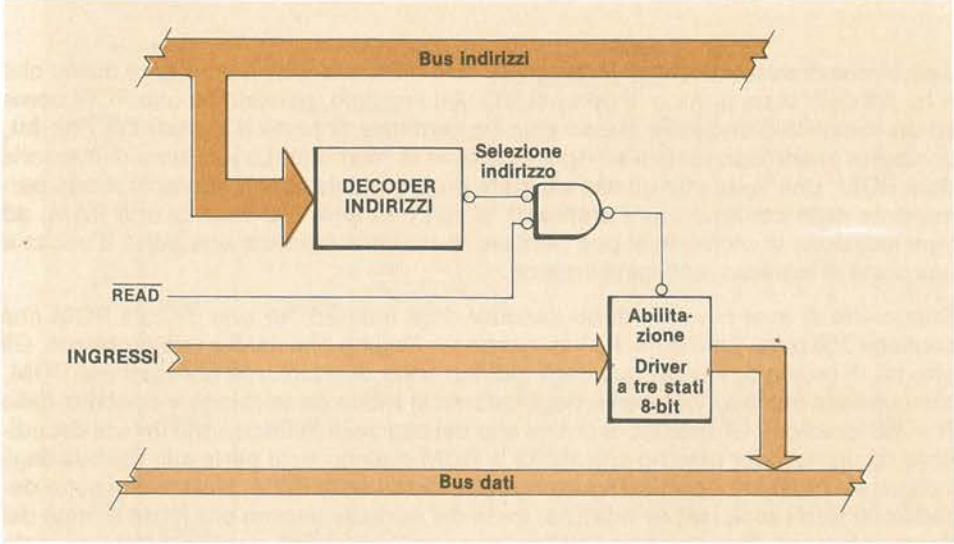


Figura 7-9. I dati d'ingresso sono posti sul bus dei dati ogni volta che il microprocessore legge l'indirizzo assegnato al driver a tre stati

DECODIFICA DEGLI INDIRIZZI PER PIÙ DISPOSITIVI

Supponete di dover decodificare gli indirizzi per controllare otto, anziché una, porta di I/O. Si potrebbero utilizzare otto decoder degli indirizzi analoghi a quello della Figura 7-7; esiste però un metodo più semplice. In Figura 7-10 è mostrato un decoder che genera i segnali di selezione per gli indirizzi 3000, 3001, 3002... 3007. Per generare questi otto indirizzi, dei 16 bit del bus degli indirizzi vengono utilizzati solo i tre bit di ordine più basso (A0, A1 e A2). I tredici bit più alti possono perciò essere decodificati da un unico circuito, simile a quello della Figura 7-7. L'uscita di tale circuito viene utilizzata per abilitare un decoder, nel nostro caso un 74LS138. Questo decoder fornisce otto uscite diverse, una per ogni possibile combinazione di A0, A1, A2. Il decoder è disabilitato (tutte le uscite sono cioè false) se i tredici bit d'indirizzo superiori non hanno il valore indicato.

Nella maggior parte dei casi per selezionare un certo dispositivo non è necessario che decodifichiate tutti i sedici bit degli indirizzi. Negli esempi sono decodificati tutti

i sedici bit per evidenziare che il sistema, se necessario, ha questa capacità di indirizzamento.

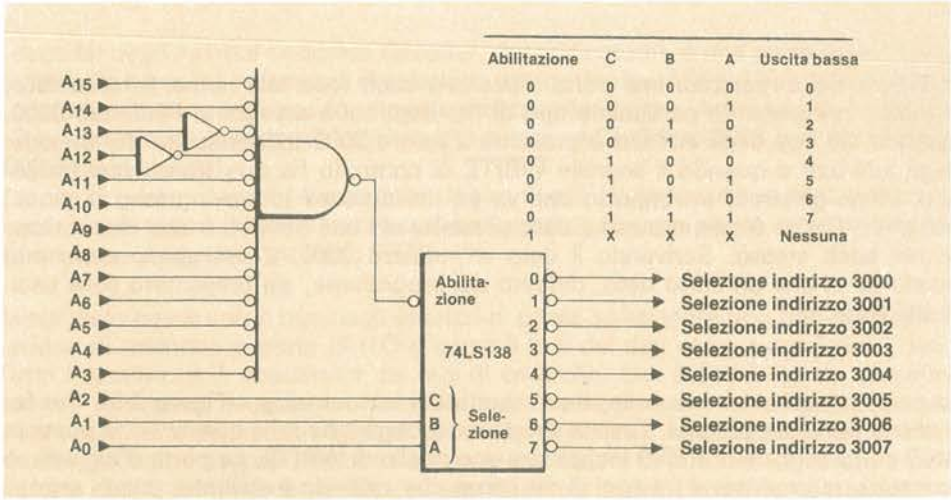


Figura 7-10. Un IC decoder permette di estendere facilmente il numero dei dispositivi che possono essere selezionati

DECODIFICA DEGLI INDIRIZZI PER LE MEMORIE

Il problema di decodificare gli indirizzi nel caso delle memorie è analogo a quello che si ha nel caso di un gruppo di porte di I/O. Ad esempio, pensate ad una ROM come ad un dispositivo che sullo stesso chip ha centinaia di porte d'ingresso di otto bit, una porta in corrispondenza ad ogni locazione di memoria. Le locazioni di memoria della ROM, una volta che questa sia stata programmata, contengono in modo permanente delle configurazioni prefissate di zeri e di uno. Nel caso di una RAM, ad ogni locazione di memoria si può pensare di far corrispondere una porta d'uscita e una porta di ingresso, collegate insieme.

Supponete di aver bisogno di un decoder degli indirizzi per una piccola ROM che contiene 256 byte. La Figura 7-11 vi mostra un circuito che risolve tale problema. Gli otto bit di ordine basso del bus degli indirizzi sono direttamente collegati alla ROM, che contiene un decoder interno degli indirizzi in modo da selezionare ciascuna delle $2^8 = 256$ locazioni. Gli otto bit di ordine alto del bus degli indirizzi sono invece decodificati da un decoder esterno che abilita la ROM quando sulla parte alta del bus degli indirizzi sia presente il particolare campo di indirizzi della ROM. Notate che sono decodificati tutti i sedici bit di indirizzo: metà dal decoder interno alla ROM e metà dal decoder esterno. Per generare l'abilitazione alla ROM, sono posti in AND il segnale di $\overline{\text{READ}}$ e l'uscita del decoder degli indirizzi. Questa tecnica è identica a quella utilizzata nel caso delle porte d'ingresso.

CONTROLLO DI PIÙ DISPOSITIVI DI MEMORIA

Poiché la maggior parte dei sistemi a microprocessore utilizzano più di un chip di memoria, è spesso necessario realizzare dei decoder d'indirizzo più complessi. Supponete di dover realizzare un sistema in cui, questa volta, siano presenti quattro ROM da 256 byte, i cui indirizzi siano quelli indicati nella Figura 7-12a.
In questa situazione le linee degli indirizzi devono indicare sia quale chip di memoria deve essere selezionato sia quale locazione deve essere indirizzata internamente al chip stesso. La Figura 7-12b riporta gli indirizzi espressi in notazione binaria, perché possiate vedere il comportamento dei singoli bit. Gli otto bit d'indirizzo più bassi definiscono la locazione all'interno del chip, mentre gli otto bit più alti definiscono quale chip si vuole indirizzare.
Notate che per decodificare un chip piuttosto che un altro viene modificato solo lo stato dei bit A8 ed A9. La ragione per cui cambiano solo questi due bit, è che per decodificare 1024 indirizzi servono esattamente 10 bit ($2^{10} = 1024$) e A9 è appunto il de-

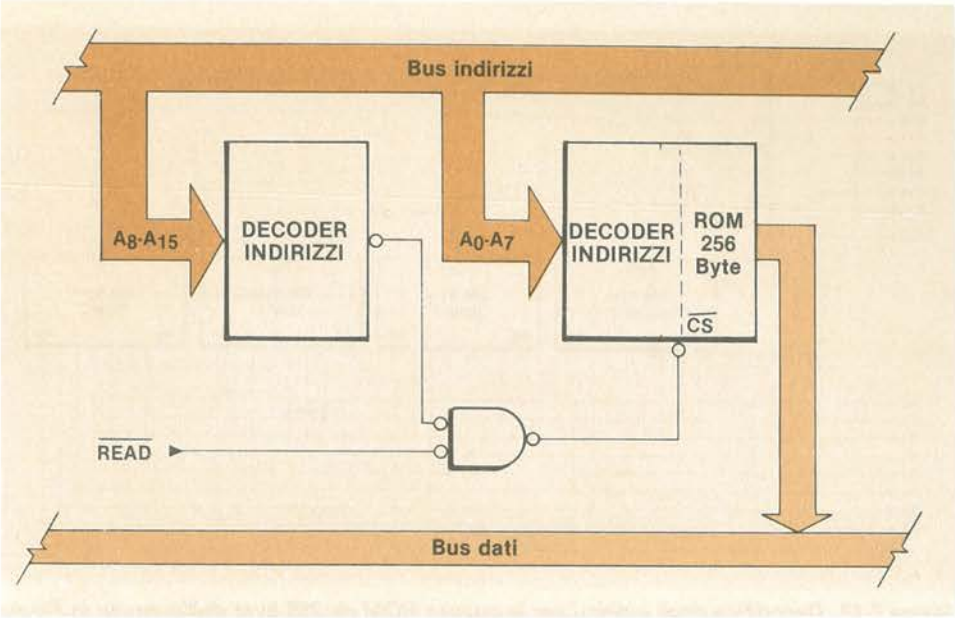


Figura 7-11. Il decoder interno alla ROM permette di ridurre il numero di linee di indirizzo richieste ad un decoder esterno

a

Indirizzo	
0	ROM 0
1	
2	
.	
.	
.	
255	
256	ROM 1
257	
.	
.	
.	
511	
512	ROM 2
513	
.	
.	
.	
766	
767	ROM 3
768	
.	
.	
.	
1023	

b

		NUMERO DELLA ROM								LOCAZIONE ALL'INTERNO DELLA ROM							
	Bit di indirizzo:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROM 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	.																
	.																
	255	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ROM 1	256	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	257	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
	.																
	.																
	511	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
ROM 2	512	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	513	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
	.																
	.																
	766	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1
ROM 3	767	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	768	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
	.																
	.																
	1023	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Figura 7-12 a e b. Indirizzi assegnati a ciascuna delle quattro ROM da 256 byte presenti in un sistema

cimo bit. Ciascuna delle quattro possibili combinazioni di A8 ed A9 definisce così un blocco di 256 indirizzi (la dimensione di ogni ROM) su quattro. Nella figura è riportato l'intero bus degli indirizzi (16 bit), poiché il microprocessore può indirizzare fino a 64K locazioni, anche se tale possibilità è usata di rado.
La Figura 7-13 vi mostra come è realizzato questo tipo di indirizzamento. Gli otto bit più bassi del bus degli indirizzi sono portati direttamente alle linee d'indirizzo di tutte e quattro le ROM, poiché tali bit specificano la locazione internamente al chip. Il decoder degli indirizzi prende invece gli otto bit più alti degli indirizzi, in modo da generare i segnali di selezione dei chip. I due bit meno significativi (degli otto più alti), A8 ed A9, sono portati agli ingressi binari del decoder, mentre gli altri bit, di ordine più elevato, fanno sì che il decoder venga abilitato solo quando essi sono tutti bassi.

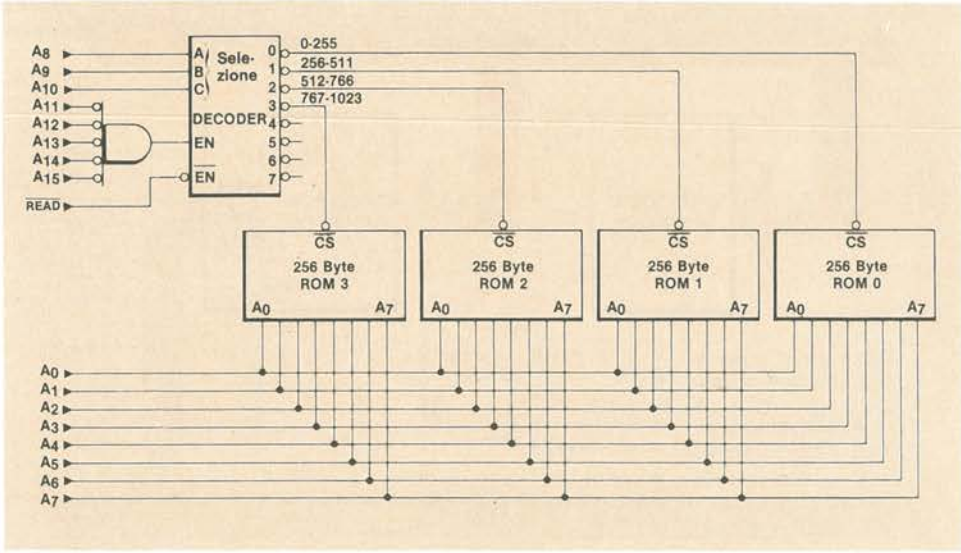


Figura 7-13. Decodifica degli indirizzi per le quattro ROM da 256 byte dell'esempio in Figura 7-12

Studiate questo schema riferendovi alla tabella (Figura 7-12b) che riporta la mappa degli indirizzi: dovreste essere in grado di osservare la corrispondenza tabella-schema. Notate che il segnale di controllo $\overline{\text{READ}}$ viene utilizzato come abilitazione del decoder, il che equivale a porre in AND tale segnale con ciascuna uscita del decoder stesso.

La tecnica di decodifica degli indirizzi presentata, a cui peraltro possono essere introdotte numerose varianti, è completa e «pulita» ed illustra molto bene i principi fondamentali: i bit di ordine basso degli indirizzi sono collegati direttamente alle linee di indirizzo delle memorie, mentre i bit di ordine alto vengono decodificati in modo da generare i segnali di selezione dei chip (chip select o chip enable).

CONTROLLO DELLA RAM

Le RAM vengono decodificate in modo analogo a quanto detto per le ROM; sono solo necessari alcuni circuiti di controllo ulteriori per poter oltre che leggere (uscita dalla RAM) anche scrivere (ingresso nella RAM). Le RAM, oltre ad un $\overline{\text{CS}}$ (Chip Select), hanno anche un ingresso di $\overline{\text{WRITE}}$ (scrittura). La Figura 7-14 vi mostra la tabella della verità (Truth Table) per controllare la RAM. Perché possa aver luogo una operazione di lettura o di scrittura è necessario che $\overline{\text{CS}}$ sia basso. Se $\overline{\text{WRITE}}$ è alto (non è vero) quando $\overline{\text{CS}}$ è basso la RAM presenta in uscita sul bus dei dati il dato, che può così essere letto dal processore: $\overline{\text{CS}}$ ha infatti abilitato i driver a tre stati delle uscite della RAM, che non vengono invece abilitati se $\overline{\text{WRITE}}$ è basso. In quest'ultimo caso i dati presenti sul bus dei dati vengono invece immagazzinati (scritti) nella locazione di memoria definitiva dal bus indirizzi.

$\overline{\text{CS}}$	$\overline{\text{WR}}$	FUNZIONE
0	0	WRITE (Scrittura)
0	1	READ (Lettura)
1	X	NESSUNA OPERAZIONE

0 = BASSO
1 = ALTO
X = NON SIGNIFICATIVO

Figura 7-14. Tabella della verità per il controllo della RAM

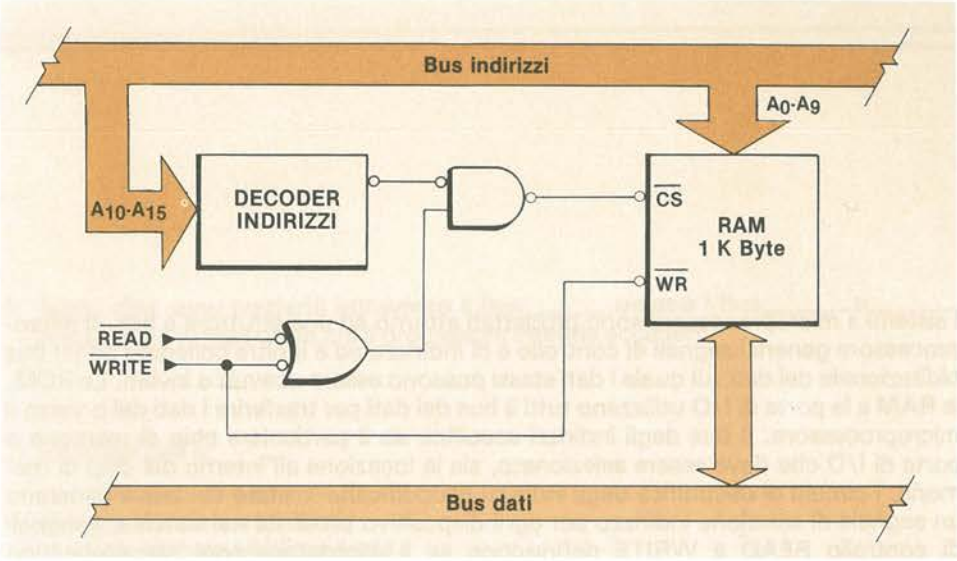


Figura 7-15. Controllo e decodifica degli indirizzi per una RAM da 1 K byte facendo uso della tabella della verità della figura 7-14

Nella Figura 7-15 è riportato un circuito che permette di realizzare la funzione appena descritta. $\overline{CS}$ è basso se sono bassi sia la selezione dell'indirizzo della RAM che il segnale di $\overline{READ}$ o $\overline{WRITE}$. La linea di $\overline{WRITE}$ è collegata direttamente all'ingresso $\overline{WRITE}$ della RAM; tale ingresso è a sua volta condizionato, interiormente al chip, dall'ingresso $\overline{CS}$ in modo da essere ignorato se $\overline{CS}$ non è basso.
La Figura 7-16 è un diagramma a blocchi del μLab: mostra le sezioni più importanti del sistema e le linee di comunicazione tra di esse, riassumendo così quanto è stato descritto in questa lezione.

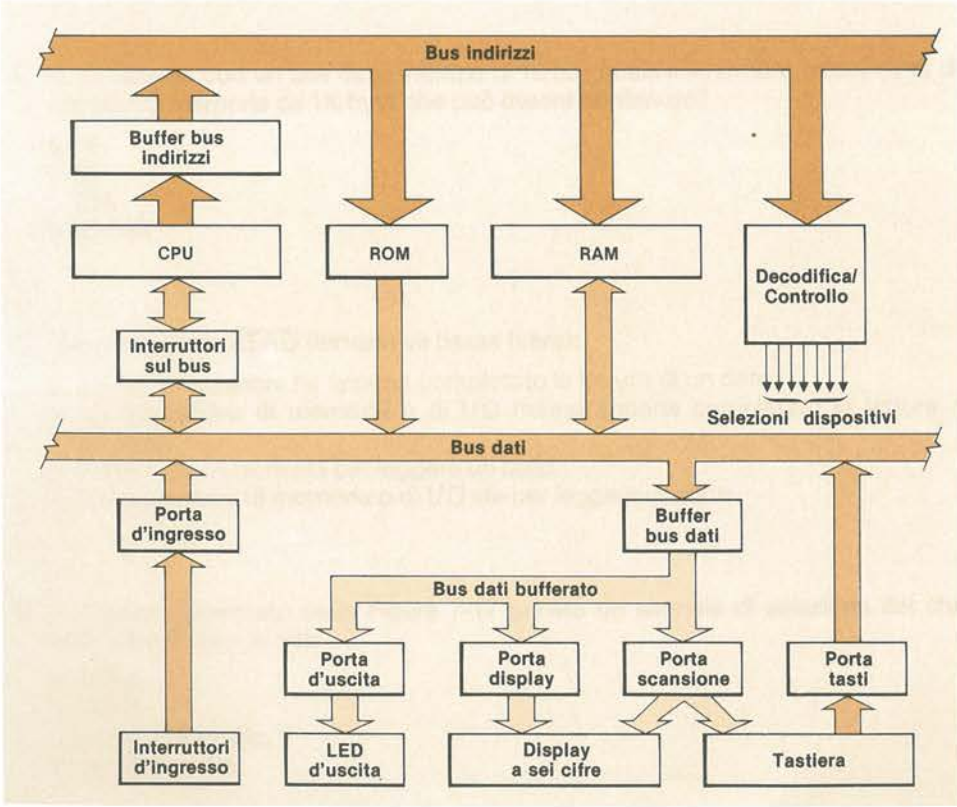


Figura 7-16. Schema a blocchi del μLab

RIPASSO

Lezione 7

I sistemi a microprocessore sono progettati attorno ad una struttura a bus. Il microprocessore genera i segnali di controllo e di indirizzo ed è inoltre collegato ad un bus bidirezionale dei dati sul quale i dati stessi possono essere ricevuti o inviati. Le ROM, le RAM e le porte di I/O utilizzano tutti il bus dei dati per trasferire i dati dal o verso il microprocessore. Il bus degli indirizzi specifica sia il particolare chip di memoria o porta di I/O che deve essere selezionato, sia la locazione all'interno del chip di memoria. I circuiti di decodifica degli indirizzi decodificano lo stato del bus e generano un segnale di selezione indirizzo per ogni dispositivo presente nel sistema. I segnali di controllo $\overline{\text{READ}}$ e $\overline{\text{WRITE}}$ definiscono se il microprocessore sta eseguendo un'operazione di lettura o un'operazione di scrittura e controllano i tempi relativi al trasferimento del dato. Il microprocessore può così controllare il modo di operare del sistema, facendo sì che tutti i dispositivi possano spartire gli stessi bus dei dati e degli indirizzi.

Lezione 7

1. Tutti i dati sono trasferiti attraverso il bus_____mentre i bus_____e_____ selezionano il dispositivo cui il processore intende accedere.

2. Il bus dei dati è:

   a. unidirezionale e a tre stati.
   b. bidirezionale e a tre stati.
   c. unidirezionale e bidirezionale.
   d. nulla di quanto detto ai punti a, b, c.

3. In un sistema che utilizza dispositivi di memoria da 1K byte, quali linee d'indirizzo saranno collegate ai chip di memoria?

   a. A0-A9
   b. A10-A15
   c. A0-A15
   d. A0-A7

4. In un sistema con un bus degli indirizzi di 16 bit, quale è il numero massimo di dispositivi di memoria da 1K byte che può essere contenuto?

   a. 16
   b. 64
   c. 256
   d. 65.536

5. Quando la linea $\overline{\text{READ}}$ (lettura) va bassa (vera):

   a. il microprocessore ha appena completato la lettura di un dato.
   b. un dispositivo di memoria o di I/O hanno appena completato la lettura di un dato.
   c. il microprocessore sta per leggere un dato.
   d. un dispositivo di memoria o di I/O sta per leggere un dato.

6. Il circuito presentato nella Figura 7-17 genera un segnale di selezione del chip (chip select) diretto alla:

   a. ROM.
   b. RAM.
   c. porta d'ingresso.
   d. porta d'uscita.

7. Il dispositivo selezionato dalla Figura 7-17 è posto all'indirizzo:
   a. A03C.
   b. 430A.
   c. 5FCC.
   d. A034.

8. Dovendo modificare il circuito della Figura 7-17 affinché possa funzionare con una porta d'uscita, uno degli ingressi di abilitazione del 74LS138 dovrebbe essere collegato a_____ anziché a_____.

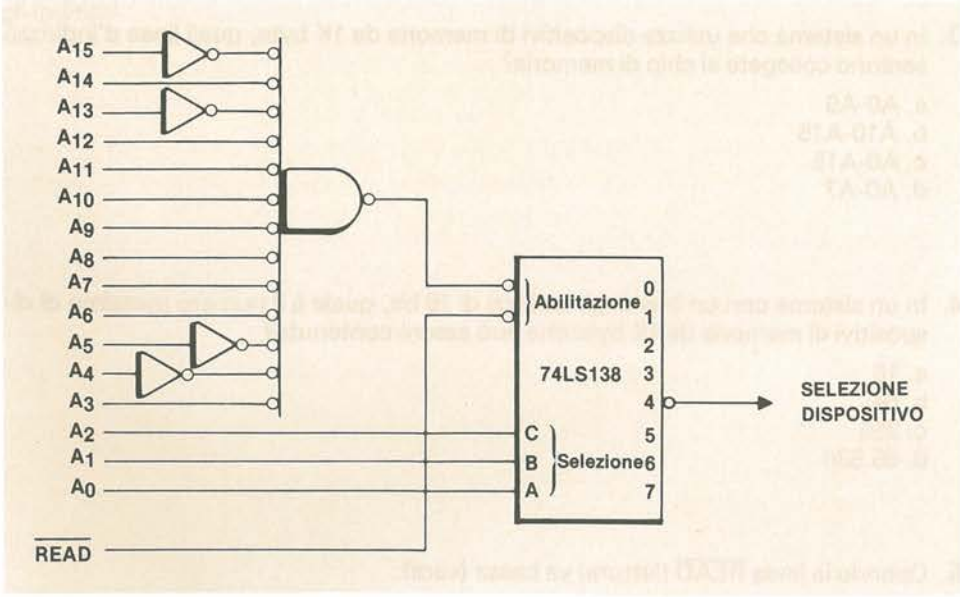


Figura 7-17. Circuito di decodifica degli indirizzi cui fanno riferimento le Domande 6, 7, 8

LEZIONE 8

Decodifica degli indirizzi

INTRODUZIONE

A tutti i dispositivi che comunicano con il microprocessore è assegnato un indirizzo particolare. I circuiti di decodifica degli indirizzi garantiscono che sul bus sia presente proprio il dispositivo indirizzato dal microprocessore.

In questa lezione vengono descritte le caratteristiche e come è fatto nel μLab il circuito di decodifica degli indirizzi. Sono anche descritti altri tipi di circuiti di decodifica degli indirizzi.

STRUTTURE D'INDIRIZZAMENTO

Nella maggior parte dei sistemi basati su microprocessore i bus dei dati e di controllo sono usati in modo abbastanza semplice e lineare, poche essendo le varianti. Numerosi sono invece i modi che si possono utilizzare per decodificare il bus degli indirizzi.

LA STRUTTURA D'INDIRIZZAMENTO DEL μLAB

Il μLab utilizza una tecnica detta I/O con *Mappa in memoria* (Memory Mapped I/O): le porte di I/O sono considerate dei dispositivi indirizzabili all'interno dell'area di memoria. Da un punto di vista software questo implica che operazioni di lettura o scrittura sulle porte possono essere effettuate servendosi di una qualunque delle istruzioni di riferimento in memoria: «MOV A,M», «MOV M,A», «STA», «LDA» ecc. (Per la descrizione di tali istruzioni, ci si riferisca alla Lezione 11). Esiste anche un'altra tecnica detta *Decodifica con mappa* in I/O (I/O Mapped Decoding) di cui si parlerà più avanti in questa lezione.

In Figura 8-1 è mostrata la *Mappa degli indirizzi* (Address Map) del μLab. È utilizzato solo il primo quarto dell'area di 64K indirizzi, e per tale motivo i bit 14 e 15 sono sempre posti a zero. Questa area di indirizzi è ancora suddivisa in otto parti uguali, ciascuna composta da 2K locazioni (0800 hex = 2.048 in notazione decimale). La ROM occupa i primi 2K indirizzi, mentre alla RAM sono assegnati i successivi 2K. La RAM ha a disposizione 2K byte di spazio di indirizzi, anche se in realtà contiene solo 1K byte. Lo spazio di 1K indirizzi che avanza non è utilizzato.

Byte più alto dell'indirizzo (in binario)								Indirizzo (in hex)				Dispositivo	
Bit: 15	14	13	12	11	10	9	8						
0	0	0	0	0	0	0	0	0	0	0	0	ROM	Memoria
0	0	0	0	0	1	1	1	0	7	F	F		
0	0	0	0	1	0	0	0	0	8	0	0	RAM	
0	0	0	0	1	1	1	1	0	F	F	F		
0	0	0	1	0	0	0	0	1	0	0	0	CONTROLLO	I/O
0	0	0	1	0	1	1	1	1	7	F	F		
0	0	0	1	1	0	0	0	1	8	0	0	TASTIERA	
0	0	0	1	1	1	1	1	1	F	F	F		
0	0	1	0	0	0	0	0	2	0	0	0	PORTA D'INGRESSO	
0	0	1	0	0	1	1	1	2	7	F	F		
0	0	1	0	1	0	0	0	2	8	0	0	SCANSIONE (Kbd + Display)	
0	0	1	0	1	1	1	1	2	F	F	F		
0	0	1	1	0	0	0	0	3	0	0	0	PORTA D'USCITA	
0	0	1	1	0	1	1	1	3	7	F	F		
0	0	1	1	1	0	0	0	3	8	0	0	SEGMENTI DISPLAY	
0	0	1	1	1	1	1	1	3	F	F	F		
0	1	0	0	0	0	0	0	4	0	0	0	NON UTILIZZATI	
1	1	1	1	1	1	1	1	F	F	F	F		

Figura 8-1. Mappa completa degli indirizzi per il μLab

Per le porte di I/O sono utilizzati i sei blocchi di 2K byte ciascuno che vengono dopo la RAM. La porta di controllo è utilizzata dal programma monitor per realizzare alcune funzioni speciali che verranno descritte nella Lezione 10. Le porte per l'ingresso dei tasti, per la scansione e per il display servono a controllare la tastiera e il display di visualizzazione, mentre le porte di ingresso e di uscita servono per gli interruttori e per i LED.

Tutte queste porte possono ricevere o inviare un solo byte di dato ciascuna, per cui ad ognuna di esse sarebbe sufficiente associare un solo indirizzo. Ad ogni porta sono invece associati 2.048 indirizzi, per cui ogni porta risponde ad uno qualunque dei suoi 2K indirizzi. Altrimenti detto, questo vuol dire che ad ogni porta sono associati 2.047 indirizzi ridondanti.

Perché utilizzare questa tecnica se di fatto sembra solo sprecare così tanti indirizzi? La ragione sta nel fatto che, così facendo, l'hardware risulta semplificato, mentre rimane comunque disponibile un'area di indirizzi ben maggiore di quanto il sistema possa richiedere. Il μLab utilizza complessivamente 16K indirizzi per la sua ROM da 2K, per la RAM da 1K e per le sei porte di I/O, mentre per una espansione del sistema rimangono ancora liberi 48K indirizzi. Il μLab avrebbe anche potuto occupare solo 2.048 + 1.024 + 6 = 3.078 indirizzi, ma unicamente a spese di un ben più complesso circuito di decodifica degli indirizzi. Questa filosofia di indirizzamento è in effetti quella più frequentemente usata nei sistemi di piccole e medie dimensioni.

L'HARDWARE DI DECODIFICA

Basterà un esame dell'hardware utilizzato per decodificare gli indirizzi, per mostrarvi quanto questo approccio abbia semplificato i circuiti. La Figura 8-2 vi mostra i circuiti di controllo e di decodifica degli indirizzi presenti nel μLab. Date un'occhiata agli indirizzi binari riportati nella Figura 8-1 e notate che sono le linee A11, A12 e A13 che specificano quale sezione, delle otto, viene indirizzata. Queste tre linee entrano, come linee binarie di selezione, nel 74LS138, decoder binario uno-a-otto: questo dispositivo ha poi otto uscite distinte, una per ciascuno dei blocchi da 2K presenti nel sistema. Con questa soluzione si ottiene un circuito di decodifica degli indirizzi relativamente semplice. La semplicità deriva direttamente dall'aver assegnato ad ogni dispositivo un blocco di indirizzi di pari lunghezza.

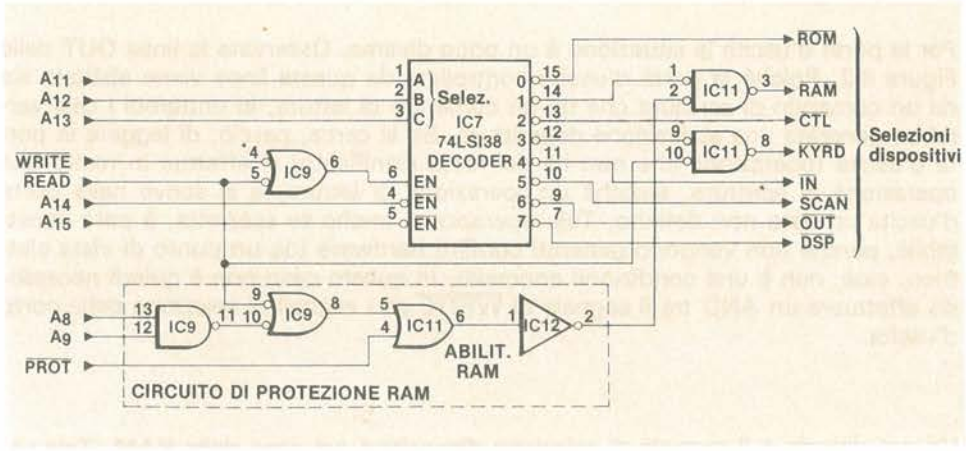


Figura 8-2. Circuito di decodifica degli indirizzi nel μLab.

Il 74LS138 ha tre ingressi di abilitazione: due attivi bassi e uno attivo alto. Tutti e tre devono essere veri perché una qualunque delle uscite possa essere a sua volta vera. Le linee A14 e A15 (collegate alle abilitazioni attive basse) impediscono che le uscite diventino vere se A14 e A15 non sono contemporaneamente bassi: i dispositivi del μLab vengono così raggruppati nei 16K indirizzi inferiori, dell'area di 64K possibili.

Collegando le linee di $\overline{\text{READ}}$ e di $\overline{\text{WRITE}}$ agli ingressi di abilitazione del decoder, si ha la garanzia che i dispositivi che utilizzano il bus saranno abilitati solo durante una operazione di lettura ($\overline{\text{READ}}$) o di scrittura ($\overline{\text{WRITE}}$) e si evita così di dover condizionare con le linee di $\overline{\text{READ}}$ e di $\overline{\text{WRITE}}$ parecchi dei segnali di selezione dispositivo. Per tale motivo il terzo ingresso di abilitazione del decoder è collegato

all'uscita di un dispositivo che effettua l'OR di $\overline{READ}$ e $\overline{WRITE}$. Così facendo le uscite di selezione dispositivo saranno vere solo quando sarà in corso un'operazione di lettura o di scrittura. Il bus degli indirizzi contiene informazioni significative solo durante questi periodi, per cui, abilitando le uscite di selezione dispositivo solo in tali tempi, si evita di leggere o scrivere dati nei dispositivi in istanti sbagliati. Questo tipo di abilitazione può anche non essere richiesto in alcuni sistemi basati su microprocessori.

C'è dell'altro. La ROM e le porte d'ingresso devono essere selezionate solo quando è in corso una lettura. Se tali dispositivi infatti rispondessero sia in caso di scrittura che di lettura, si potrebbe verificare un *conflitto sul bus.* Ad esempio, se si effettua un'operazione di scrittura verso la ROM, il microprocessore porrà dei dati sul bus dei dati; dato che la scrittura in ROM è impossibile, se durante l'operazione di scrittura è abilitata, la ROM stessa cercherà di porre dei dati sul bus dei dati. Si verifica così una soluzione inaccettabile che potrebbe causare dei danni elettrici ai circuiti.

Questo problema viene risolto ponendo in AND il segnale di $\overline{READ}$ con il segnale selezione dispositivo della ROM: è sufficiente a tal fine collegare il $\overline{READ}$ ad uno degli ingressi di abilitazione della ROM. Sul μLab la ROM ed i chip delle porte d'ingresso hanno a disposizione due piedini d'abilitazione; su di uno viene riportato il segnale di abilitazione del dispositivo, mentre sull'altro è portato il $\overline{READ}$. Questo modo di utilizzare i due piedini di enable è del tutto equivalente ad effettuare l'AND tra il segnale di $\overline{READ}$ ed il segnale di selezione dispositivo. (Si veda lo schema generale del circuito nel risvolto di copertina).

In Figura 8-2 è mostrato come quest'ultima soluzione è utilizzata per controllare la porta d'ingresso della tastiera (KYRD). Poiché la porta della tastiera non ha un ulteriore piedino di selezione, ci si serve del dispositivo IC11C per realizzare la funzione logica AND.

Per le porte d'uscita la situazione è un poco diversa. Osservate la linea OUT della Figura 8-2. Poiché la porta d'uscita controllata da questa linea viene abilitata sia da un comando di scrittura che da un comando di lettura, in entrambi i casi verrebbe generata una abilitazione di scrittura. Se si cerca, perciò, di leggere la porta d'uscita (operazione che non ha del resto significato) si effettua in realtà una operazione di scrittura, anziché un'operazione di lettura, e si scrive nella porta d'uscita un dato non definito. Tale operazione, anche se scorretta, è però accettabile, perché non vengono generati conflitti hardware (da un punto di vista elettrico, cioè, non è una condizione anomala). In questo caso non è quindi necessario effettuare un AND tra il segnale di $\overline{WRITE}$ ed i segnali di selezione delle porte d'uscita.

IL CIRCUITO DI PROTEZIONE DELLA RAM

Un po' diverso è il segnale di selezione dispositivo nel caso della RAM. Tale segnale deve essere vero quando nell'area di indirizzi RAM sono in corso o una lettura o una scrittura. La linea di selezione dispositivo della RAM è invece condizionata dal circuito di protezione dalla scrittura (Write Protect Circuit) che è descritto nei paragrafi seguenti.

Il circuito di *protezione dalla scrittura* serve a diminuire la possibilità che venga accidentalmente distrutto il contenuto della RAM. Può capitare che a causa di un errore di programmazione, anche abbastanza banale, il microprocessore esegua delle sequenze di operazioni non corrette, in pratica che stia girando in modo non controllato (spesso interpretando dei dati come se fossero delle istruzioni). Questo errore comporta spesso la memorizzazione in tutta la RAM di dati del tutto casuali, con il risultato di distruggere il programma contenuto in RAM.

Per prevenire tale problema, il μLab contiene un latch detto *protezione della memoria*. L'uscita di questo latch corrisponde al segnale $\overline{PROT}$ indicato nel circuito

della Figura 8-2. Quando il latch è posto a uno, la RAM è protetta: si può leggere ma non scrivere in RAM. Il monitor porta automaticamente a uno il latch di protezione ogni volta che voi fate partire un programma; quando invece non sta girando un vostro programma, porta il latch a zero, così da permettervi di introdurre dati o modificare programmi.

Poiché poi, durante l'esecuzione dei programmi, vi può capitare di voler memorizzare in RAM dei dati, vengono protetti solo i primi tre quarti della RAM stessa. Le linee A8 e A9 definiscono quale quarto della RAM è indirizzato: quando entrambe le linee sono alte, vuol dire che è indirizzato l'ultimo quarto della RAM che non deve essere protetto. Le linee A8 e A9 sono così poste in AND; il risultato di tale AND è poi posto in OR con il segnale di $\overline{\text{READ}}$ e l'uscita, ancora, è posta in OR con $\overline{\text{PROT}}$. Quest'ultima uscita genera il segnale di abilitazione della RAM.

L'esperimento seguente (e gli altri di questa sezione) fa uso di un oscilloscopio al fine di mostrarvi alcuni segnali del μLab. L'oscilloscopio deve avere una banda passante di almeno 10 MHz ed essere a doppia traccia. Per permettervi di verificare la correttezza delle vostre operazioni, tutto quanto dovreste vedere visualizzato sullo schermo è riportato anche nelle diverse figure. Le stesse, nel caso non aveste a disposizione un oscilloscopio, vi sono utili per comprendere comunque l'esperimento.

Nelle discussioni che seguono, un trattino separa i numeri dei piedini di ogni IC dal numero del circuito integrato stesso. Così il piedino 18 del circuito integrato IC13 è indicato con IC13-18. I numeri dei circuiti integrati sono riportati sulla scheda del circuito stampato.

ESPERIMENTO 8-1

Il decoder degli indirizzi

INTRODUZIONE

In questo esperimento introdurrete e farete girare un programma che abiliterà in sequenza tutti i dispositivi indirizzabili presenti nel μLab. Utilizzando un oscilloscopio potrete osservare i diversi segnali di selezione dispositivo.

PROCEDIMENTO

A) Introducete in tastiera il seguente programma:

Indirizzo	Contenuto	Label	Istruzione		Commenti
0800	3A	LOOP	LDA	0000	;ROM
0801	00				
0802	00				
0803	32		STA	1000	;Porta di controllo
0804	00				
0805	10				
0806	3A		LDA	1800	;Porta tasti
0807	00				
0808	18				
0809	3A		LDA	2000	;Porta d'ingresso
080A	00				
080B	20				
080C	32		STA	2800	;Porta di scansione
080D	00				
080E	28				
080F	32		STA	3000	;Porta d'uscita
0810	00				
0811	30				
0812	32		STA	3800	;Porta display
0813	00				
0814	38				
0815	C3		JMP	LOOP	;Ripeti
0816	00				
0817	08				

B) Collegate la sonda del canale A dell'oscilloscopio al foro di prova di selezione ROM (indicato con $\overline{ROM}$), posto al di sotto dei LED del bus degli indirizzi sul μLab.

C) Collegate il canale B alla linea di selezione RAM (IC7-14). Fate attenzione a non fare dei corti tra i piedini, in caso contrario potreste perdere il vostro programma. È utile porre una clip di test sull'IC7.

D) Mettete entrambi gli ingressi sulla scala 2V/div e impostate la velocità di scansione su 10 μs/div. Sincronizzatevi sul canale A.

E) Fate partire il programma che avete introdotto nel passo A. Sullo schermo dell'oscilloscopio si dovrebbe vedere una immagine simile a quella della Figura 8-3. La ROM è abilitata solo una volta per ogni loop (ogni loop dura circa 50 μs). La RAM è abilitata invece ogni volta che viene letto un byte d'istruzione. Notate che quando il segnale di selezione della ROM è basso, la RAM non è abilitata (il segnale di selezione della RAM non è basso). Notate ancora che ci sono altri sei periodi, all'interno del loop, durante i quali la RAM non è abilitata: sono i periodi durante i quali sono selezionati gli altri sei dispositivi.

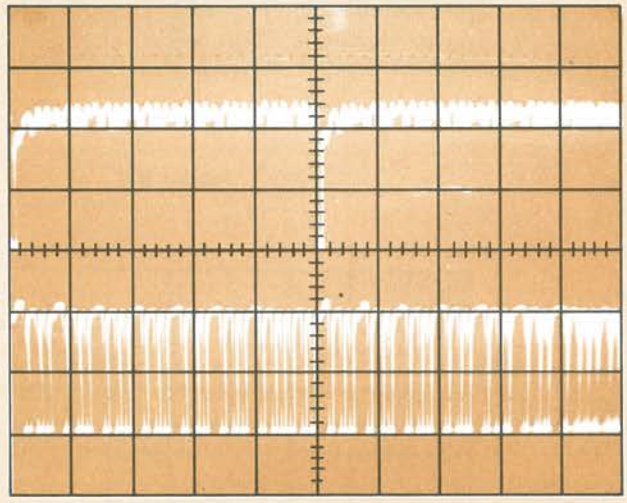


Figura 8-3. La RAM è abilitata (segnale basso) ogni volta che in questo loop di programma viene letto un byte di istruzione

F) Spostate la sonda B sugli altri piedini di abilitazione (IC7) ed osservate (Figure da 8-4 a 8-9) che l'ordine in cui sono abilitati gli altri dispositivi corrisponde alla sequenza delle istruzioni del programma. In nessun istante due dispositivi (RAM compresa) sono contemporaneamente abilitati.

ESPERIMENTO 8-1

(continuazione)

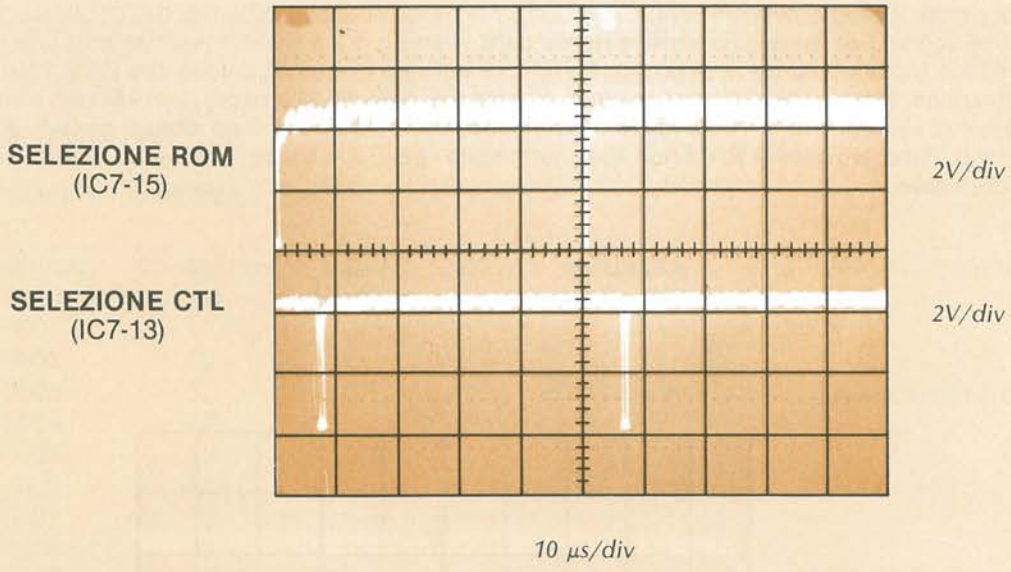


Figura 8-4.

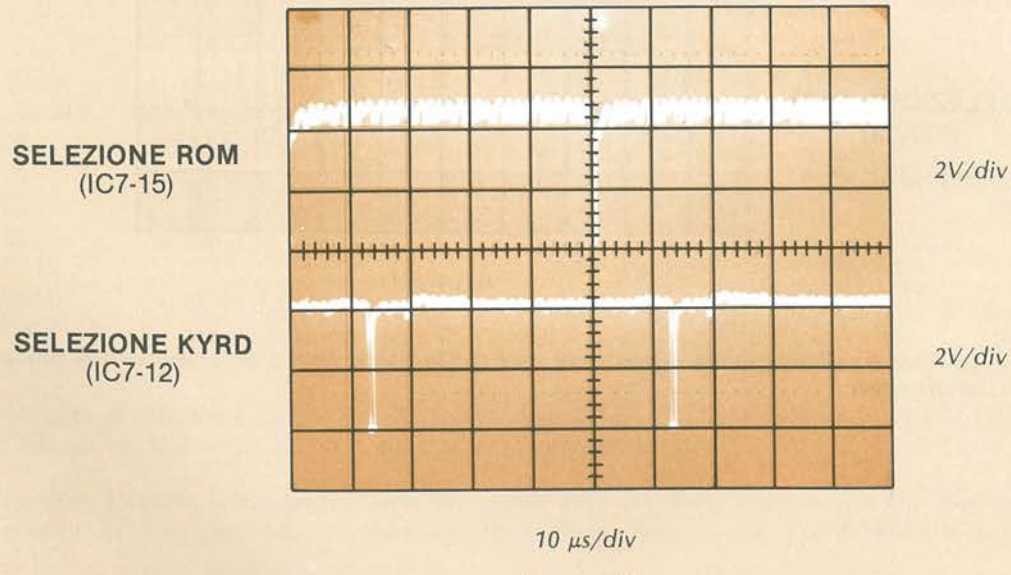


Figura 8-5.

Figure da 8-4 a 8-9. L'ordine secondo cui i dispositivi vengono selezionati segue l'ordine delle istruzioni del programma

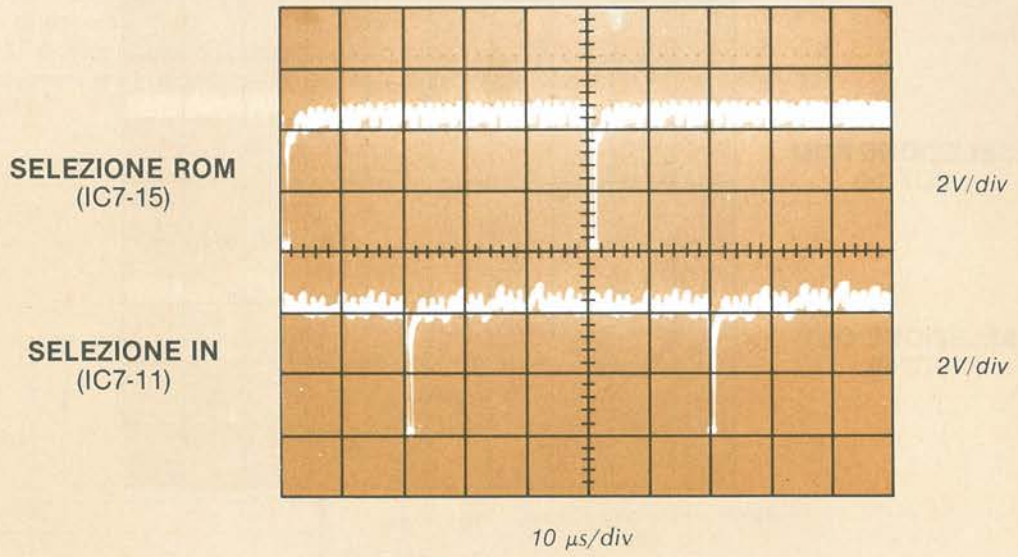


Figura 8-6.

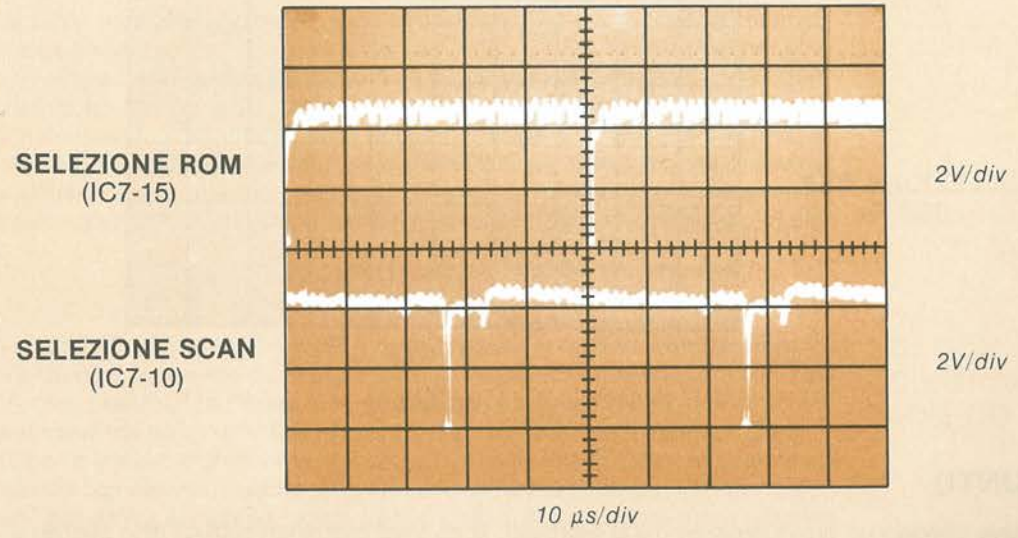


Figura 8-7.

ESPERIMENTO 8-1

(continuazione)

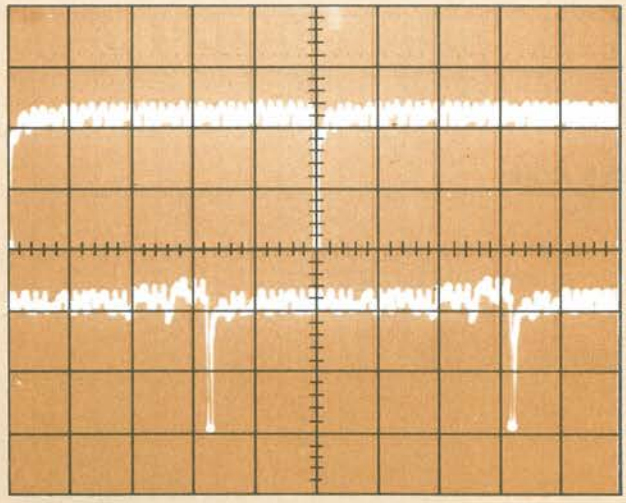


Figura 8-8.

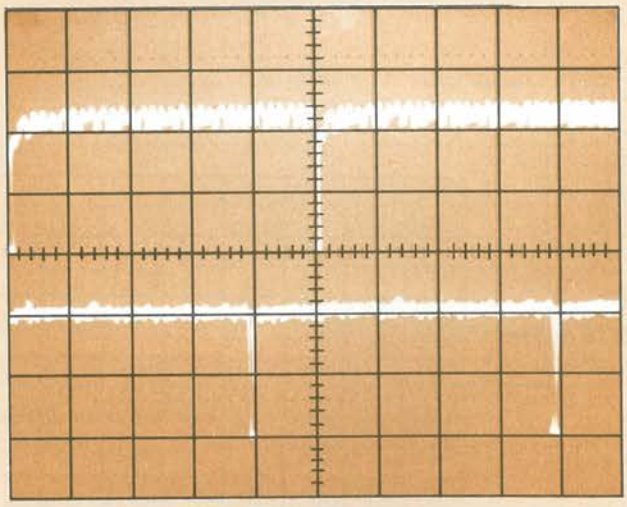


Figura 8-9.

RIASSUNTO

Mentre stava girando un breve programma d'esercizio, sono stati esaminati tutti gli otto segnali di selezione dispositivo. È stato così facile osservare come la sequenza delle istruzioni del programma controllava le linee di abilitazione dispositivo.

ALTRE TECNICHE DI DECODIFICA

Esistono alcuni altri modi per decodificare gli indirizzi. Per ogni applicazione, la scelta della tecnica da usare è legata a parecchi fattori: la capacità di memoria, il numero delle periferiche, la necessità di prevedere un'espansione del sistema e i tipi specifici di dispositivi di memoria utilizzati. In alcuni casi questi dispositivi contengono più piedini di abilitazione, che possono quindi essere usati per realizzare parte della decodifica degli indirizzi.

DECODIFICA CON SELEZIONE LINEARE

La più semplice di tutte le tecniche di decodifica, in cui non si usa nessuna logica di decodifica degli indirizzi, è la *selezione lineare*. I bit più alti degli indirizzi vengono direttamente utilizzati come segnali di selezione dei chip. Nella Figura 8-10 è riportato un esempio di decodifica con selezione lineare. La RAM viene selezionata tutte le volte che A15 è alto, in corrispondenza cioè di tutti gli indirizzi da 8000 a FFFF. La ROM è selezionata invece quando A14 è alto, condizione che è vera per gli indirizzi da 4000 a 7FFF.

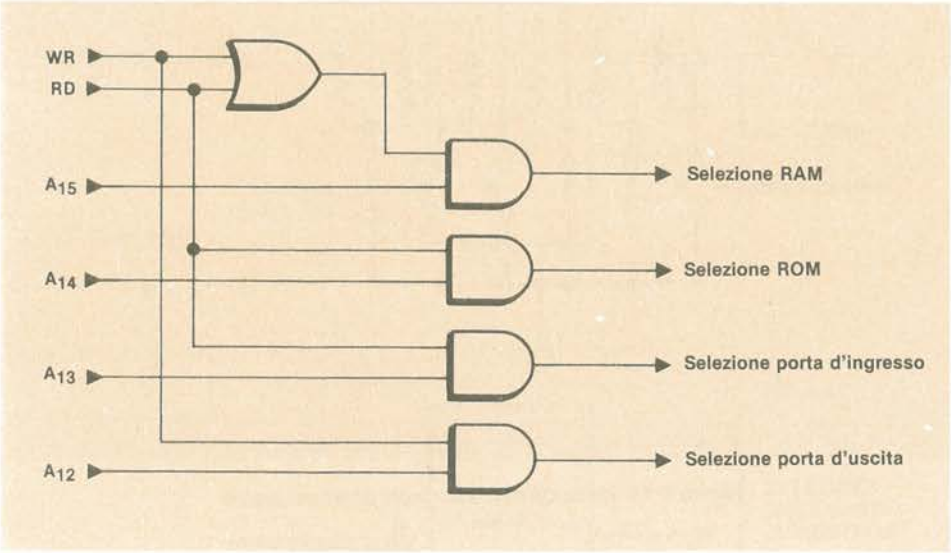


Figura 8-10. Decodifica con selezione lineare

Notate che la ROM è selezionata anche quando sono alti A14 e A15 contemporaneamente, in corrispondenza, cioè, degli indirizzi da C000 a FFFF, sovrapponendosi in questo campo all'area della RAM. Se si cerca di effettuare una lettura in uno di questi indirizzi, vengono abilitati tutti e due i dispositivi e si verifica perciò un conflitto sul bus. È questo uno degli svantaggi della decodifica lineare: a causa di tale problema il software deve evitare di leggere tutti quegli indirizzi in cui sia vero più di uno dei bit più significativi. Un ulteriore svantaggio di questa soluzione è che viene sprecata una grande parte dell'area di memoria, per cui tale soluzione è utilizzata solo in piccoli sistemi.

DECODIFICA CON COMPARATORE LOGICO

Una delle tecniche più eleganti e flessibili è quella che si serve di un *comparatore logico*: con tale tecnica, partendo da N ingressi d'indirizzo, viene selezionata una sola area tra le 2^N aree possibili. In Figura 8-11 è mostrato un circuito che genera un segnale di selezione dispositivo utilizzando i sei bit di ordine alto degli indirizzi. Ogni ingresso A del comparatore è confrontato al rispettivo ingresso B. Quando tutte le sei coppie di ingressi coincidono, l'uscita del comparatore si porta bassa. Gli interruttori sono utilizzati per definire il livello logico degli ingressi B del comparatore. Questa tecnica è particolarmente utile per schede di memoria e di periferiche: in esse saranno presenti interruttori o ponticelli, in modo da definire l'indirizzo di ogni scheda all'interno di un sistema. Per realizzare la funzione di confronto, si possono usare anche dei componenti Exclusive - OR.

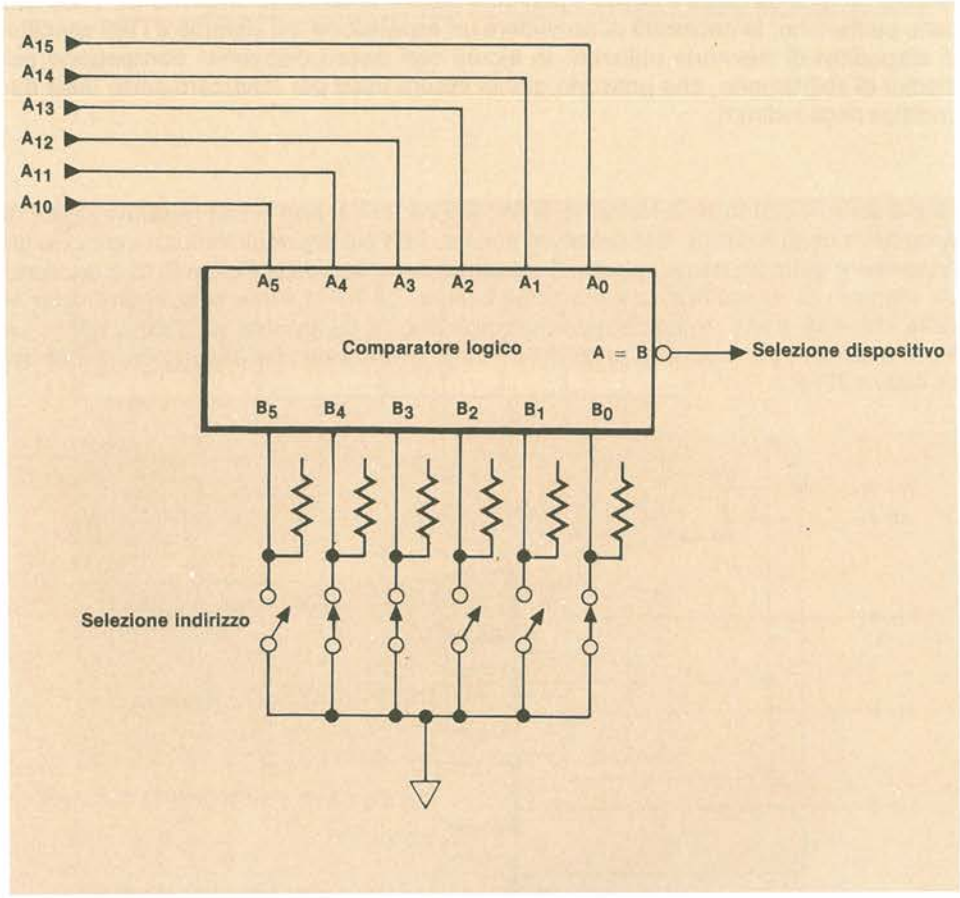


Figura 8-11. Decodifica con un comparatore logico

DECODIFICA CON LOGICA COMBINATORIA

In sistemi che richiedono ben poche linee decodificate sono spesso usati dei componenti logici standard (gate). La Figura 8-12 mostra un componente NAND a quattro ingressi, preceduto da invertitori, che decodifica gli indirizzi da 9000 a 9FFF. L'uscita si porta bassa tutte le volte che le linee di indirizzo da A15 a A12 si trovano rispettivamente nello stato 1,0,0,1. Completando o non completando gli ingressi di indirizzo del NAND, si possono generare abilitazioni diverse per sedici (2^4) dispositivi.

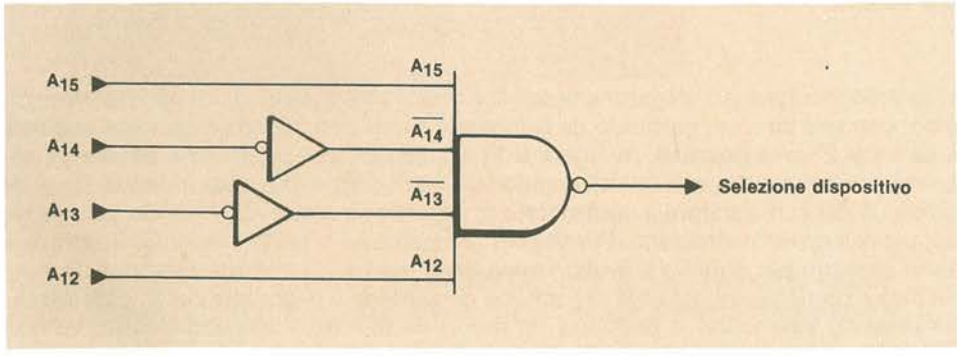


Figura 8-12. Decodifica con un dispositivo logico NAND

DECODIFICA CON MAPPA IN I/O

Alcuni microprocessori (e tra questi i microprocessori 8080 e 8085) dispongono di una ulteriore linea di controllo che specifica se l'indirizzo è da associare alla memoria o ai dispositivi di I/O. Nel microprocessore 8085 questa linea è denominata IO/$\overline{M}$ (si veda la Figura 8-13) ed è controllata dall'istruzione che è in corso di esecuzione. Durante tutte le operazioni di trasferimento in memoria, IO/$\overline{M}$ è bassa. Quando viene eseguita una delle due istruzioni di I/O (IN o OUT), la linea IO/$\overline{M}$ diventa alta e abilita la parte di I/O. Con IO/$\overline{M}$ bassa è invece abilitata la memoria. Utilizzando questo metodo la memoria e i dispositivi di I/O hanno aree d'indirizzo diverse: viene così aumentata (di $2^8 = 256$ byte) l'area totale che può essere indirizzata, ed è così permessa una maggiore flessibilità di progetto delle decodifiche. Inoltre le istruzioni IN e OUT sono istruzioni a due byte (specificano una delle 256 porte di I/O). Per ogni trasferimento in dispositivi di I/O viene così risparmiato un byte di memoria di programma rispetto alle istruzioni a tre byte che fanno riferimento alla memoria (ad esempio «LDA» e «STA»).

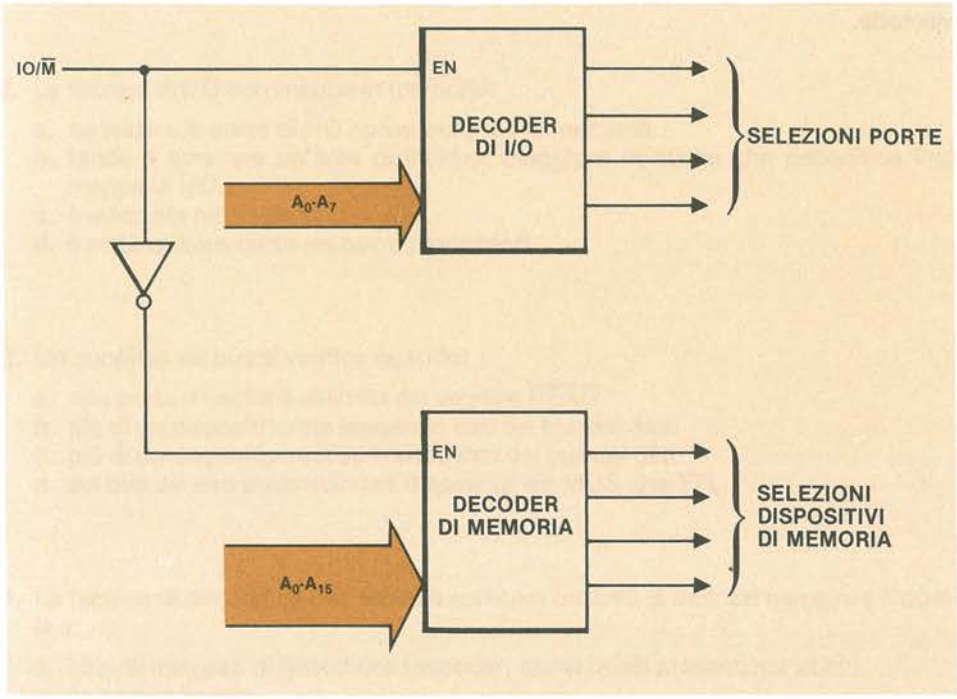


Figura 8-13. La decodifica con mappa di I/O aumenta di 256 byte l'area indirizzabile

Poiché il μLab non fa uso della mappa in I/O per le sue porte di I/O, la linea IO/$\overline{M}$ non è utilizzata. Il μLab, invece, utilizza la tecnica dell'I/O con mappa in memoria (Memory Mapped I/O) già discussa in questa lezione.

RIPASSO

Lezione 8

Nei sistemi basati su microprocessori, per decodificare gli indirizzi, sono utilizzate parecchie tecniche. Per tali circuiti sono spesso utilizzati circuiti integrati di decodifica (decoder), comparatori e componenti standard (gate). Nei sistemi piccoli il decoder degli indirizzi è in genere più semplice, a spese però di un maggior spreco dell'area d'indirizzamento disponibile. Una tecnica comunemente usata è quella dell'I/O mappato in memoria, in cui le porte di I/O sono indirizzate come locazioni di memoria.

Lezione 8

1. Ogni dispositivo in un sistema a microprocessore:
   a. ha associato un certo indirizzo specifico.
   b. risponde sempre ad un unico indirizzo.
   c. è sempre controllato da un circuito di decodifica degli indirizzi.
   d. è tutto quanto detto nei punti precedenti.

2. La tecnica di I/O con mappa in memoria:
   a. considera le porte di I/O come locazioni di memoria.
   b. tende a sprecare un'area di indirizzi maggiore di quella che decodifica una mappa di I/O.
   c. è utilizzata nel μLab.
   d. è tutto quanto detto nei punti precedenti.

3. Un conflitto sul bus si verifica quando:
   a. una porta d'uscita è abilitata dal segnale $\overline{\text{READ}}$.
   b. più di un dispositivo sta leggendo dati dal bus dei dati.
   c. più di un dispositivo sta scrivendo dati dal bus dei dati.
   d. sul bus dei dati sono utilizzati dispositivi sia MOS che TTL.

4. La tecnica di decodifica che tende a sprecare un'area di indirizzi maggiore è quella a:
   a. circuiti integrati di decodifica (decoder, come quelli presenti nel μLab).
   b. selezione lineare.
   c. compatori logici.
   d. logica combinatoria.

5. Se dei dispositivi vengono abilitati quando non sono presenti né un $\overline{\text{READ}}$ né un $\overline{\text{WRITE}}$:
   a. si possono verificare dei conflitti sul bus.
   b. possono essere persi i dati contenuti nella ROM.
   c. non si ha nessun problema poiché questo è uno stato normale.
   d. nessuno dei punti a, b, c, è vero.

6. Si può indirizzare un maggior numero di locazioni di memoria e di I/O utilizzando la seguente tecnica di decodifica:
   a. selezione lineare.
   b. I/O mappato in memoria.
   c. decodifica con mappa in I/O.
   d. decodifica con comparatore logico.

LEZIONE 9
Memorie e periferiche

INTRODUZIONE

Se, di un sistema, il microprocessore può essere considerato il cuore, il bus dei dati diventa il flusso sanguigno e le periferiche e le memorie gli organi. In questa lezione vengono descritte le memorie e le periferiche che sono presenti nel μLab e in altri sistemi.

MEMORIE

Due sono i tipi, ben distinti, di memorie a semiconduttore disponibili: RAM e ROM. Numerose sono poi le varianti che si possono avere all'interno di ciascuno di questi due tipi fondamentali. In questo capitolo descriveremo alcuni dispositivi di memoria tra i più comuni.

RAM

In quasi tutti i microcalcolatori, le RAM sono utilizzate sia come memorie «programmabili» dall'utente sia per la memorizzazione di dati. Esistono due tipi di RAM: statiche e dinamiche. Nelle RAM statiche ogni elemento di memoria è costituito da un flip-flop; un circuito integrato di RAM da 1K bit contiene quindi 1024 flip-flop. Si può comandare ogni flip-flop in modo tale che memorizzi uno zero od un uno. I circuiti di decodifica interni al chip di RAM selezionano, tra tutti quelli presenti, il particolare flip-flop indicato dalle linee di indirizzo. Lo stato del flip-flop non viene alterato, a meno che non vi venga memorizzato un nuovo dato o cada l'alimentazione della RAM.

Le RAM dinamiche utilizzano invece, come elemento di memorizzazione, un condensatore. In generale la presenza di carica sul condensatore indica uno stato 1 mentre l'assenza di carica uno stato 0. Questa tecnica rende più semplice la cella di memoria e permette di realizzare dei chip di memoria con un maggior numero di elementi. Si ha però un problema: il condensatore tende a perdere la carica e dopo pochi millisecondi un uno può diventare uno zero. È necessaria perciò un'operazione che consiste nel *rinfrescare*, cioè nel leggere, entro un certo tempo, una sequenza di locazioni della RAM. Nel corso dell'operazione di lettura del dato il chip di RAM provvede automaticamente a riscrivere lo stesso dato nella locazione letta e, di conseguenza, tutti i bit a uno vengono correttamente riportati alla condizione di piena carica presente, mentre i bit a zero a quella di carica assente. Tipicamente le memorie dinamiche vengono rinfrescate almeno ogni due millisecondi.

Per rinfrescare continuamente le memorie dinamiche è necessario aggiungere dei circuiti particolari, con un conseguente aumento di complessità del sistema. Per questo motivo in sistemi piccoli si tende ad usare le memorie statiche, più semplici. Le memorie dinamiche, tuttavia, hanno alcuni vantaggi: costano meno, a pari dimensione, di quelle statiche, e di solito consumano anche meno. In genere poi le RAM più grosse (quelle cioè che contengono un maggior numero di bit) sono quelle dinamiche. Per tali motivi nei sistemi che hanno bisogno di molte memorie sono utilizzate le memorie dinamiche che consumano meno e sono più economiche. Di solito i chip di memorie dinamiche contengono un bit per ogni indirizzo di locazione (sono «larghe» un bit). Ancora, alcuni microprocessori contengono internamente al chip dei circuiti di rinfresco che semplificano l'uso delle RAM dinamiche.

La RAM del Microprocessor Lab

Il μLab utilizza RAM statiche da 4K bit (2114 o 4045), che sono organizzate come 1K × 4. Per ottenere una parola da otto bit ne sono quindi necessarie due: con due chip da 1K × 4 si ottiene infatti una RAM da 1K byte (1K × 8).

Nella figura 9-1 è mostrato il circuito della RAM del μLab. I piedini di controllo e degli indirizzi sono collegati, per entrambi i chip, alle stesse linee del bus. Le linee 0-3 dei dati sono collegate a IC5, le linee 4-7 sono collegate a IC6. I due chip formano così 1K byte di memoria.

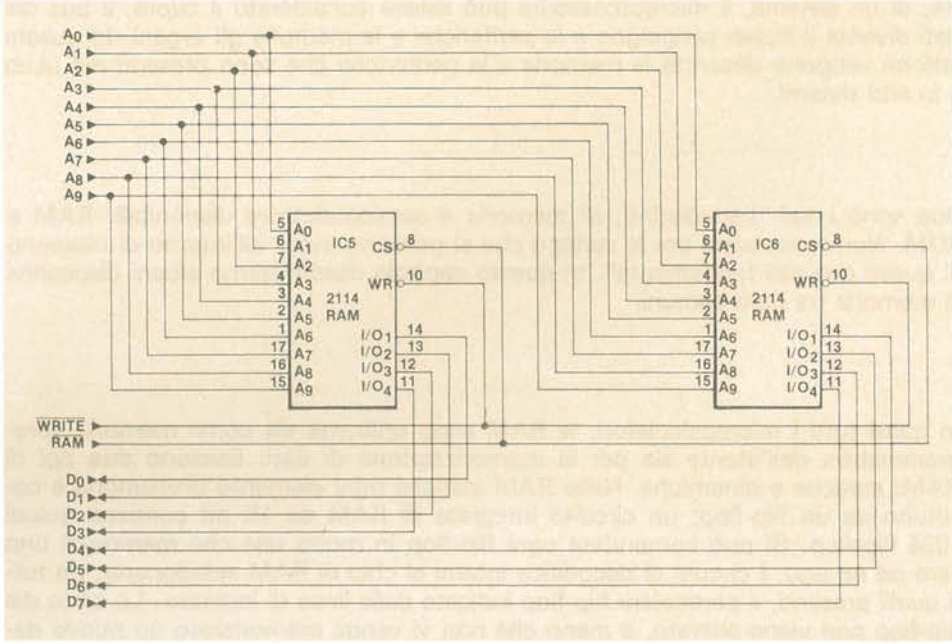


Figura 9-1. Circuito RAM del μLab

Altre configurazioni di memoria

Parecchi chip di memoria sono «larghi» un solo bit. La prima memoria statica poco costosa è stata la 2102, organizzata come 1K × 1. Poiché in ogni chip viene quindi letto o scritto un solo bit per volta, per poter leggere o scrivere 1K byte di dati sono necessari otto chip.

La Figura 9-2 vi presenta una memoria di 1K × 8 realizzata utilizzando le 2102. Come per il circuito con le 2114, le linee di controllo e degli indirizzi sono collegate, per tutti i chip, alle stesse linee dei bus rispettivi. Ad ogni chip è associato un bit del bus dei dati. La 2102 ha un piedino per l'uscita e uno per l'ingresso del dato, una soluzione che è comoda quando si hanno sistemi che non utilizzano un bus dei dati bidirezionale. Nella maggior parte delle applicazioni con microprocessori

questi piedini sono semplicemente cortocircuitati. Il circuito realizzato con le 2102 è funzionalmente equivalente a quello della Figura 9-1 solo che richiede otto IC anziché due: si ha così un chiaro esempio di come la densità dei chip di memoria stia continuamente aumentando.

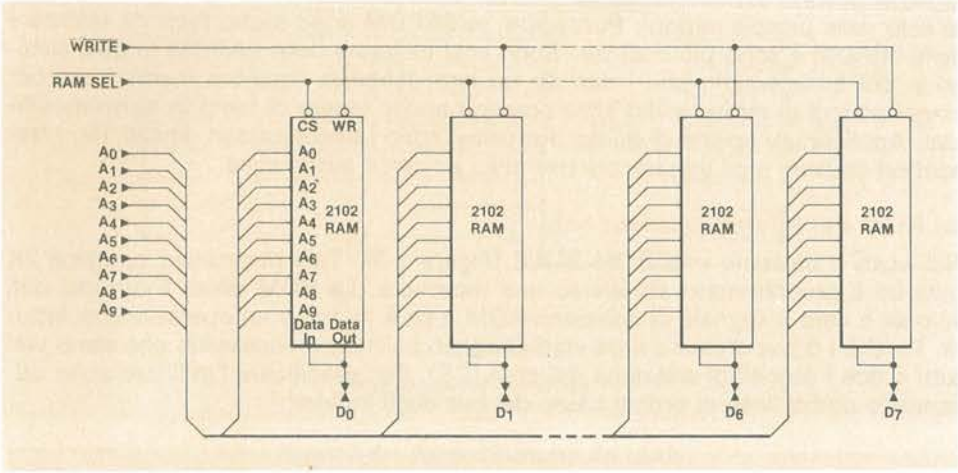


Figura 9-2. Memoria da 1K × 8 realizzata con RAM 2102 (1K × 1)

Altre memorie assai diffuse sono la 2141 (4K × 1 statica), la 2104 (4K × 1 dinamica) e la 2116 (16K × 1 dinamica). Sono sufficienti otto chip 2116 per avere 16K byte di memoria: usando le 2102 sarebbero stati necessari 128 chip!

MEMORIE A SOLA LETTURA (ROM)

Le memorie a sola lettura (ROM, Read-Only memory) danno la possibilità di memorizzare in modo permanente dati e programmi. Il programma monitor del μLab è, per esempio, memorizzato in una ROM ed è perciò sempre disponibile. Per memorizzare programmi in modo permanente non sono molto utili invece le RAM, che perdono il loro contenuto quando cade la tensione di alimentazione. La maggior parte delle ROM, utilizzate in sistemi a microprocessore, sono larghe otto bit.

Esistono quattro differenti tipi di ROM. Le *ROM programmate da maschera* (Mask-Programmed) sono programmate dal costruttore di circuiti integrati che fabbrica, in pratica, un chip «custom», dedicato cioè al cliente. Si ha di solito una spesa iniziale, dovuta al processo di fabbricazione della maschera specializzata al programma del cliente ma, da tale momento in poi, le ROM vengono a costare relativamente poco. Essendo le meno costose e quelle che offrono le maggior densità di bit immagazzinati, le ROM programmate attraverso maschera sono spesso utilizzate in prodotti di cui debbano essere realizzate quantità abbastanza grosse.

Il secondo tipo è costituito dalle *memorie programmabili a sola lettura* (PROM, Programmable Read-Only Memory). L'utilizzatore può programmare elettricamente tali ROM servendosi di uno strumento particolare detto programmatore di PROM (PROM Programmer). Tali memorie, una volta che siano state programmate, non possono essere più modificate.

Simili alle PROM, sono le *memorie programmabili cancellabili a sola lettura* (EPROM, Erasable Programmable Read-Only Memory), che possono però essere anche cancellate e riprogrammate. I bit programmati sono immagazzinati come cariche in condensatori a perdita quasi nulla. La cancellazione è effettuata esponendo a luce ultravioletta l'IC che è racchiuso in un contenitore dotato di una finestrella trasparente. Questi tipi di dispositivi sono utili quando si abbiano da realizzare prototipi o produzioni di piccola serie. Durante lo sviluppo dei μLab, per esempio, è stata utilizzata una EPROM (2716).

L'ultimo tipo di ROM ad essere introdotto è la *memoria a sola lettura alterabile elettricamente* (EAROM, Electrically Alterable Read-Only Memory). Le EAROM possono essere cancellate elettricamente senza che sia necessario toglierle dal circuito. Un vantaggio rispetto alle EPROM è costituito dal fatto che mentre le EPROM devono essere cancellate in toto, delle EAROM possono essere cancellate solo delle piccole sezioni. Purtroppo, le EAROM sono meno facili da utilizzare delle EPROM e sono più costose. Sono così utilizzate delle EAROM in quei sistemi in cui è necessario che i dati da un lato debbano rimanere memorizzati per lunghi periodi di tempo e dall'altro possano anche essere di tanto in tanto modificati. Applicazioni tipiche di questi dispositivi sono i sintonizzatori digitali TV, i trasduttori calibrati e gli apparecchi telefonici a ricerca automatica.

La ROM del Microprocessor Lab

Nel μLab è presente una ROM 2316E (Figura 9-3). Tale dispositivo contiene 2K byte ed è programmato attraverso una maschera. La ROM pilota il bus dei dati solo se è vero il segnale di selezione ROM e se è in corso un'operazione di lettura. Perché i driver d'uscita a tre stati vengano abilitati, è necessario che siano veri tutti e due i segnali di selezione del chip (CS). Per specificare l'indirizzo sono utilizzate le undici linee di ordine basso del bus degli indirizzi.

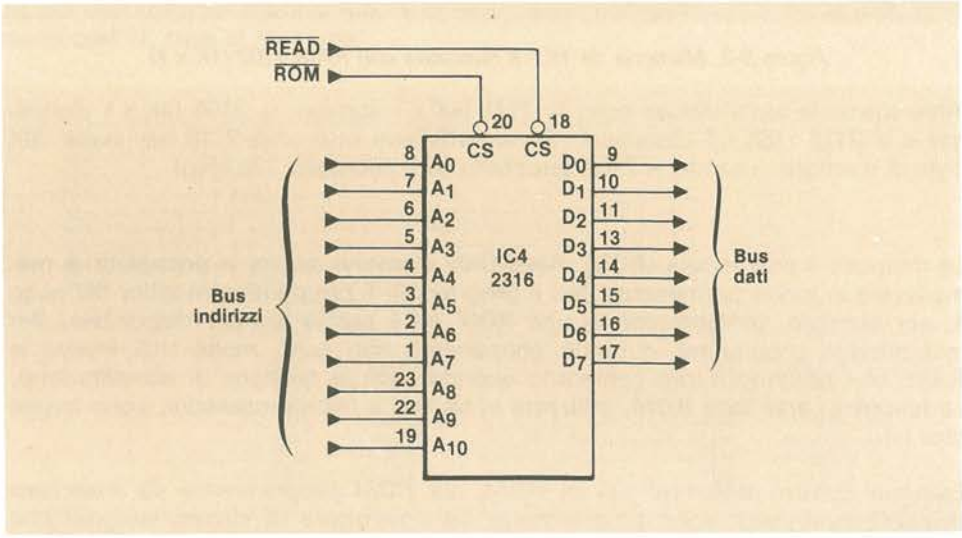


Figura 9-3. Circuito ROM del μLab

PERIFERICHE DEL MICROCALCOLATORE

Perché possa essere di una qualche utilità, un sistema a microprocessore deve poter in qualche modo interagire col mondo esterno. Deve così avere almeno un dispositivo d'ingresso ed uno d'uscita, collegati al sistema a microprocessore attraverso delle porte di I/O. Poiché non sono direttamente coinvolti nelle operazioni effettuate dal microprocessore, tali dispositivi sono detti dispositivi periferici, o *periferiche* (Peripherals).

INGRESSI ED USCITE

Il μLab dispone di due periferiche assai semplici: gli interruttori a slitta della porta d'ingresso ed i LED della porta d'uscita. Gli interruttori posti sugli ingressi della porta d'ingresso fanno sì che, se un interruttore è chiuso, l'ingresso corrispondente sia posto basso; nel caso di un interruttore aperto, invece, l'ingresso corrispondente è portato alto dalla resistenza di richiamo (Figure 9-4a e 9-4b).

Ogni linea d'uscita della porta d'uscita pilota direttamente un LED, con resistenze poste in serie al fine di limitare la corrente. Quando l'uscita è bassa, la corrente passa attraverso il LED ed il LED stesso si accende.

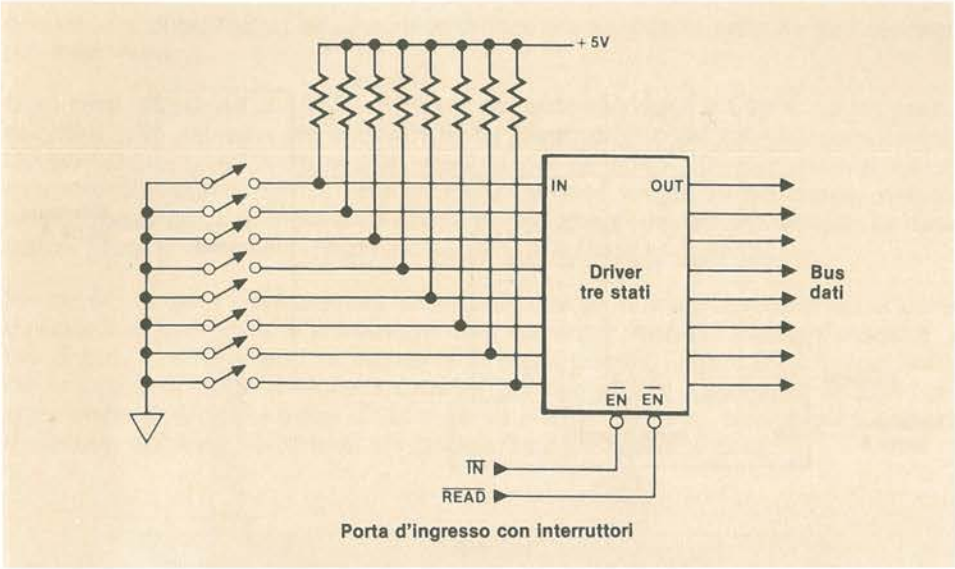


Figura 9-4a. Porta d'ingresso del μLab

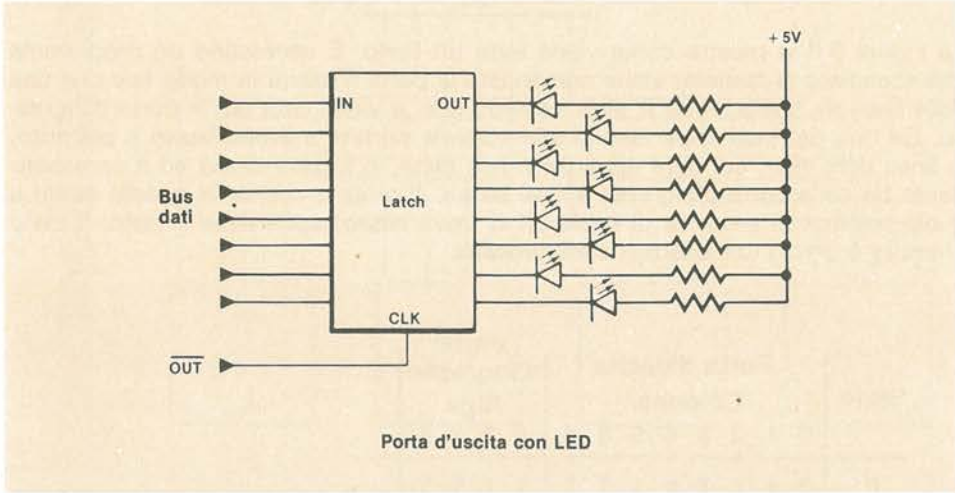


Figura 9-4b. Porta d'uscita del μLab

TASTIERA E DISPLAY

Benché interruttori e LED siano dispositivi semplici e funzionali, in numerose applicazioni è necessario che l'operatore possa disporre di qualcosa di più facile utilizzo. Nei sistemi a microprocessore, come periferiche sono assai diffuse le tastiere ed i display, che sono infatti presenti anche nel μLab. Descriveremo perciò di seguito come operano questi due tipi di periferiche.

La tastiera

In entrambi questi due tipi di periferiche è utilizzata una importante tecnica hardware detta scansione (Scanning). La tastiera del μLab è costituita da ventisei tasti. Se ogni tasto fosse direttamente collegato ad una linea della porta d'ingresso, sarebbero necessarie ben quattro porte (otto ingressi ciascuna). Servendosi di una tecnica di scansione, invece, sono sufficienti due sole porte per potersi interfacciare con un massimo di 256 tasti.

La Figura 9-5 vi presenta una interfaccia per tastiera. I tasti sono organizzati in forma di matrice, in modo da essere dei punti di collegamento tra diverse righe e colonne della matrice. Una porta d'uscita pilota le colonne, mentre le righe vengono lette da una porta d'ingresso.

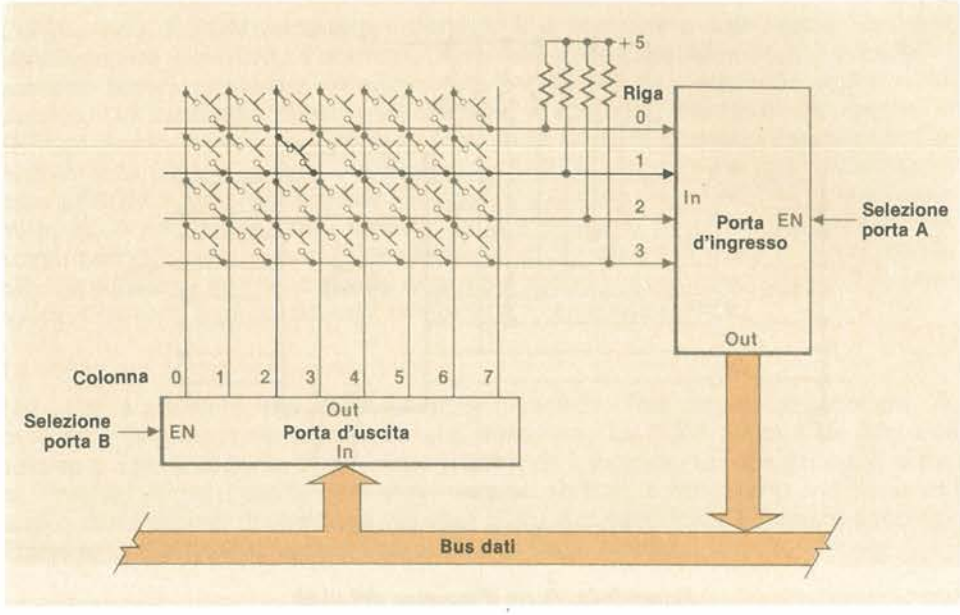


Figura 9-5. Interfaccia tastiera simile a quella del μLab

La Figura 9-6 vi mostra come viene letto un tasto. È necessario un programma che scandisca la tastiera: viene comandata la porta d'uscita in modo tale che una delle linee sia bassa, tutte le altre essendo alte, e viene letta poi la porta d'ingresso. Se uno dei tasti appartenenti alla colonna portata a livello basso è premuto, la linea della riga, cui pure appartiene tale tasto, è forzata bassa ed il corrispondente bit della porta d'ingresso è così basso. Il programma sa in questo modo a quale colonna (a seconda di quale bit si trova basso) appartiene il tasto: il tasto premuto è perciò univocamente identificato.

Stato	Porta d'uscita Colonna								Porta d'ingresso Riga			
	0	1	2	3	4	5	6	7	0	1	2	3
0	0	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1
2	1	1	0	1	1	1	1	1	1	0	1	1
3	1	1	1	0	1	1	1	1	1	1	1	1
4	1	1	1	1	0	1	1	1	1	1	1	1
5	1	1	1	1	1	0	1	1	1	1	1	1
6	1	1	1	1	1	1	0	1	1	1	1	1
7	1	1	1	1	1	1	1	0	1	1	1	1

Il tasto premuto si trova nella riga 1, colonna 2

Figura 9-6. Dati di scansione quando è premuto il tasto indicato nella Figura 9-5

Per scandire tutti i tasti, viene posto basso un bit dopo l'altro della porta d'uscita ed è effettuata la scansione dei tasti. Questa operazione è tuttavia così veloce, che tutta la tastiera sarà controllata, quattro tasti per volta, in un tempo praticamente trascurabile rispetto alla massima velocità di battitura di un operatore.

Il display

Anche per il display ci si serve di una tecnica di scansione. Benché il μLab possa visualizzare sei cifre, solo una cifra per volta è accesa. Le cifre sono accese in se-

quenza una dopo l'altra, ad una velocità tale che sembrano tutte accese contemporaneamente.

Ogni cifra (Digit) del display è formata da sette segmenti a LED e da un punto decimale (che equivale alla virgola). Sono presenti un collegamento comune per l'alimentazione e dei collegamenti separati per gli otto LED. Per visualizzare un carattere bisogna portare il collegamento comune ad un livello basso, mentre vanno portati alti i collegamenti dei segmenti che si desiderano accesi. Le resistenze poste in serie con i segmenti servono a limitare la corrente.

Per poter pilotare i display senza dover utilizzare un numero spropositato di porte d'uscita e di resistenze di limitazione della corrente, vengono collegati insieme, a mò di bus, i collegamenti di segmenti di tutti i display (Figura 9-7). È così sufficiente una sola porta d'uscita a otto bit per comandare i segmenti di tutti i display presenti. Un'altra porta d'uscita serve poi per pilotare i collegamenti comuni di ciascun display e definire quindi quale cifra deve essere accesa.

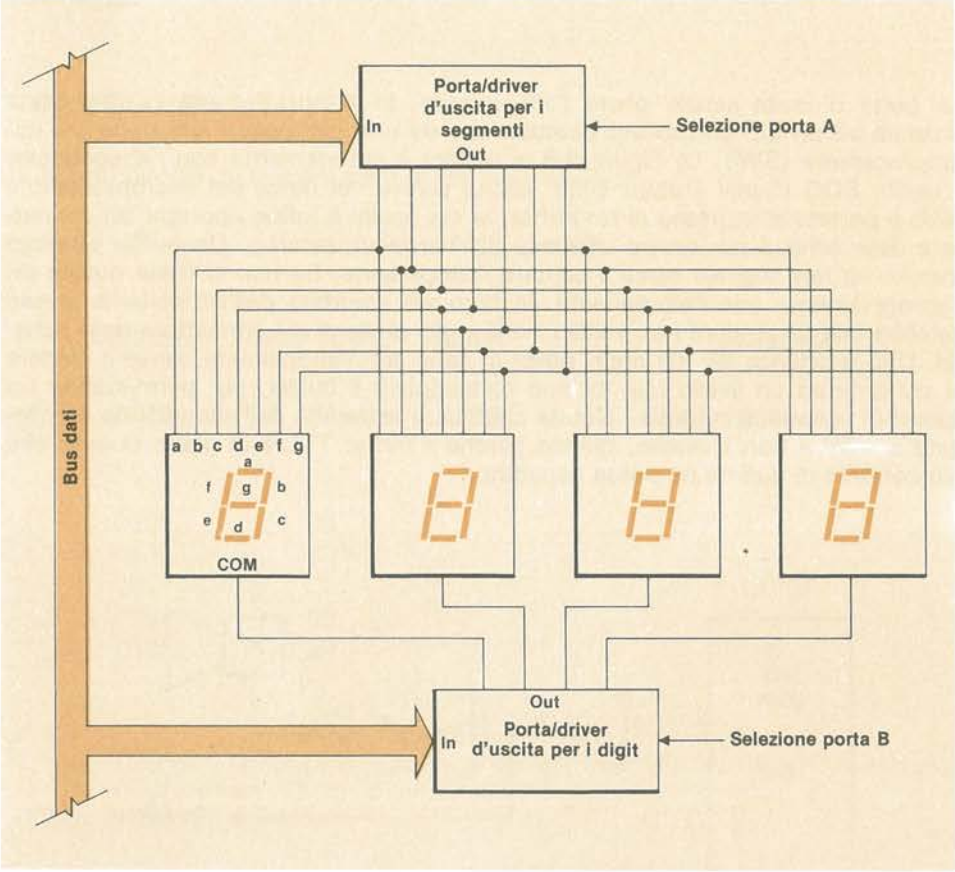


Figura 9-7. Interfaccia display simile a quella del μLab

Per controllare il display serve naturalmente un po' di software. Per prima cosa deve essere inviata alla porta dei segmenti l'informazione relativa ai segmenti della cifra zero. A questo punto sulla porta delle cifre (detta anche dei digit) viene scritta una configurazione che faccia diventare attiva la sola cifra zero. Quando è stata accesa per un certo intervallo di tempo (controllato da un loop di temporizzazione presente nel programma), la cifra zero viene spenta (attraverso la porta delle cifre). Può ora essere inviata alla porta dei segmenti l'informazione relativa ai segmenti del digit uno e tale cifra può essere abilitata dalla porta dei digit. Questo processo viene continuamente ripetuto, essendo sempre accesa solo una cifra per volta.

La soluzione descritta è un eccellente esempio di come è possibile sostituire l'hardware con del software. Prima, di solito ad ogni cifra era associato un circuito di pilotaggio con parecchi componenti di controllo e di pilotaggio. Un altro compito del software di controllo display è quello di definire, dato un certo carattere da visualizzare, quali sono i segmenti che devono essere accesi. Fino ad ora ci si serviva a tal fine di un decoder sette-segmenti per ogni cifra. Oltre ad eliminare la presenza del decoder, il software permette anche di poter costruire dei caratteri nuovi, a piacimento. Il software presente nella memoria del microprocessore vi permette di visualizzare facilmente qualunque configurazione desideriate, basta solo che sia compatibile con il formato sette segmenti. Questa possibilità è utilizzata da alcuni programmi dimostrativi della ROM del μLab per generare alcuni particolari caratteri.

Questi circuiti sono in effetti molto simili a quelli presenti veramente nel μLab. La differenza sta nel fatto che nel μLab la porta di scansione è utilizzata contemporaneamente come porta delle colonne per la tastiera e come porta delle cifre per il display (tecnica questa spesso utilizzata per ottimizzare l'hardware).

LA PORTA D'USCITA SERIALE

La porta d'uscita seriale pilota l'altoparlante. In effetti si tratta di una porta d'uscita ad un bit, controllata direttamente da una particolare istruzione del microprocessore (SIM). La Figura 9-8 vi mostra il collegamento con l'altoparlante. L'uscita SOD (Serial Output Data, uscita seriale del dato) del microprocessore 8085 è portata all'ingresso di un buffer, la cui uscita è infine riportata sul connettore della scheda per essere utilizzata dall'hardware esterno. Un buffer ulteriore inserito su tale segnale serve a pilotare l'altoparlante. Se non ci fosse questo disaccoppiamento con l'altoparlante, la corrente assorbita dall'altoparlante stesso sarebbe tale da rendere non validi i livelli logici presenti sul connettore della scheda. Una resistenza da 100 ohm, posta in serie con l'altoparlante, serve a limitare la corrente ad un livello tale da non danneggiare il buffer, pur permettendo un suono di volume accettabile. Notate che l'altra estremità dell'altoparlante è collegata a +5V e non a massa, questo perché il buffer TTL è in grado di assorbire più corrente di quanta ne possa generare.

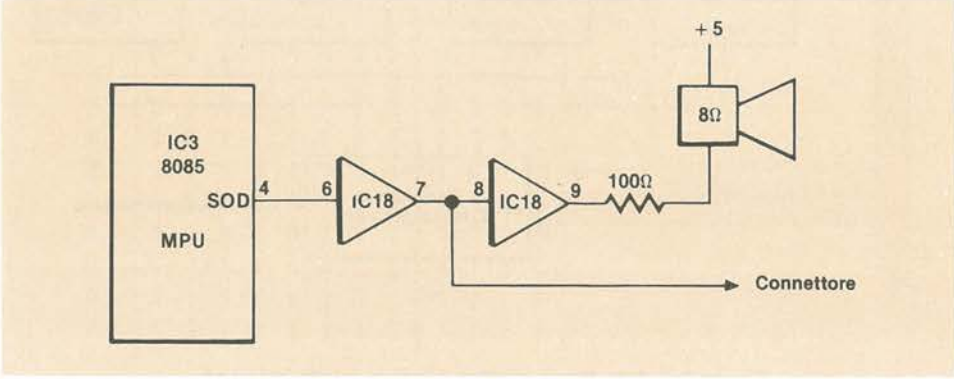


Figura 9-8. Circuito di uscita seriale del μLab.

Il software controlla la frequenza della nota che viene generata. Il programma «BEEP» contenuto nella ROM commuta parecchie centinaia di volte al secondo l'uscita seriale tra il livello alto e il livello basso, pilotando così l'altoparlante con un'onda quadra.

ESPERIMENTO 9-1

Esecuzione dei programmi

INTRODUZIONE

Durante lo svolgimento di questo esperimento, mentre nel μLab starà girando un breve programma, esaminerete con un oscilloscopio le uscite del decoder degli indirizzi. Potrete in questo modo seguire la sequenza delle operazioni. Esaminerete inoltre tali segnali anche mentre sarà in esecuzione un programma lungo.

PROCEDIMENTO

A) Introducete da tastiera il seguente programma:

Indirizzo	Contenuto	Label	Istruzione		Commenti
0800	32	LOOP:	STA	3000	;Copia l'accumulatore nella porta d'uscita.
0801	00				
0802	30				
0803	C3		JMP	LOOP	;Ripeti.
0804	00				
0805	08				

B) Collegate la sonda del canale A dell'oscilloscopio alla linea di selezione porta d'uscita del decoder degli indirizzi (IC7-9). State attenti a non cortocircuitare dei piedini; potreste altrimenti cancellare il vostro programma.

C) Collegate il canale B alla linea di selezione della RAM (IC7-14).

D) Regolate entrambi i canali a 2V/div con una velocità di scansione di 2 μs/div e sincronizzatevi sul canale A.

E) Fate partire il programma che avete introdotto nel passo A. Verificate che sullo schermo sia presente un'immagine simile a quella riportata nella Figura 9-9. Ricordatevi che entrambi i segnali sono attivi bassi. Il segnale di selezione della RAM si porta basso ad ogni riferimento in memoria, ovvero (in questo programma) ogni volta che viene letto un altro byte del programma. Notate che mentre la RAM è selezionata sei volte (il programma infatti è costituito da sei byte), la porta d'uscita è invece selezionata una sola volta (quando il programma scrive un dato sulla porta).

ESPERIMENTO 9-1

(continuazione)

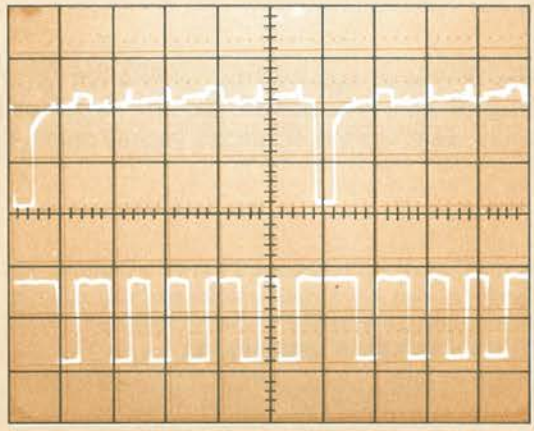


Figura 9-9. Segnali di selezione chip mentre è in esecuzione un breve programma a loop

F) Quanto tempo dura un loop del programma? (Risposta: 11,5 μs).

G) Notate che i due dispositivi (RAM e porta d'uscita) non sono mai selezionati contemporaneamente. Esaminate gli altri piedini di selezione dispositivi del circuito IC7. Poichè il programma non fa riferimento ad altri dispositivi, tutti gli altri segnali di selezione rimangono alti.

H) Premete [RESET] in modo da ritornare al programma monitor del μLab.

I) Servendovi di un solo canale dello oscilloscopio, esaminate le linee di selezione della ROM e della RAM (IC7, piedini 15 e 14). Regolate il canale d'ingresso a 1V/div per una velocità di 1 μs/div. Le Figure 9-10 e 9-11 vi mostrano alcune immagini tipiche. Poiché il programma che sta girando (il monitor) effettua un loop assai lungo, è difficile (e talvolta impossibile) sincronizzarsi sulle forme d'onda. In effetti la forma d'onda è periodica, ma i periodi sono molto lunghi. Vi possono essere utili, se presenti nell'oscilloscopio, gli ingressi di mantenimento (Trigger Holdoff) e di controllo ritardo (Delay Control).

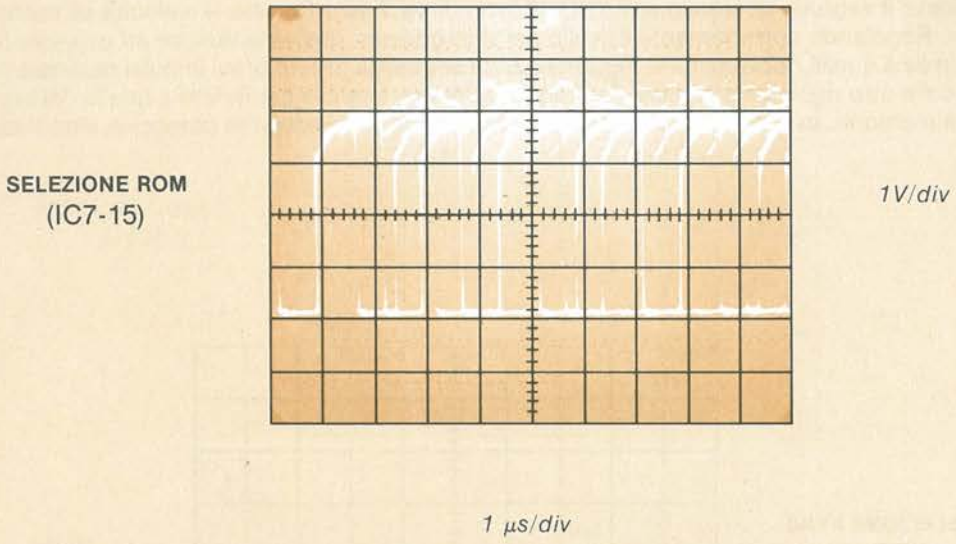


Figura 9-10. Segnale di selezione ROM mentre sta girando il programma monitor

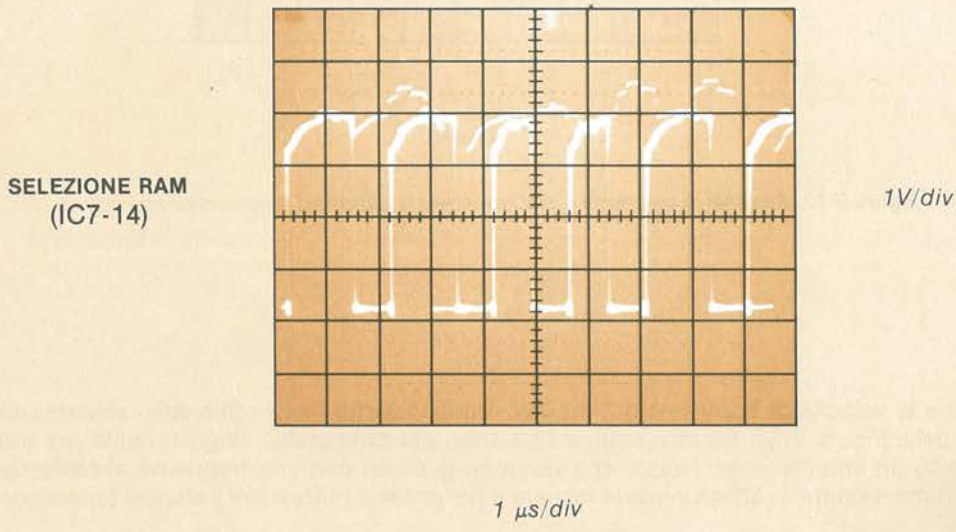


Figura 9-11. Segnale di selezione RAM mentre sta girando il programma monitor

(continuazione)

J) Esaminate il segnale di selezione $\overline{\text{KYRD}}$ (Keyread) (IC7-12). Portate la velocità di scansione a 50 μs/div. Regolando correttamente il livello del sincronismo, dovreste riuscire ad ottenere una immagine simile a quella riportata nella Figura 9-12. La sequenza di otto brevi impulsi rappresenta la scansione delle otto righe della tastiera. La frequenza di tale segnale è inferiore a quella dei segnali di selezione memoria, in quanto tra due letture della tastiera sono eseguite parecchie altre letture in memoria.

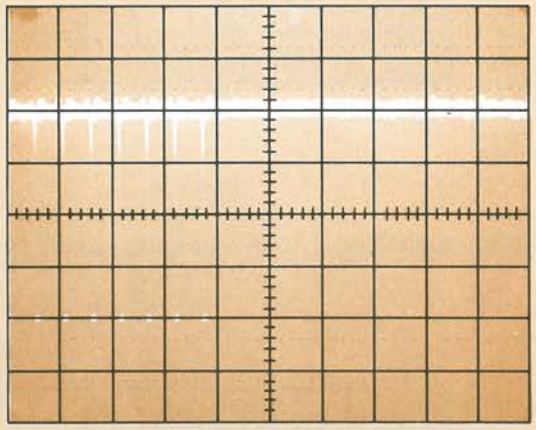
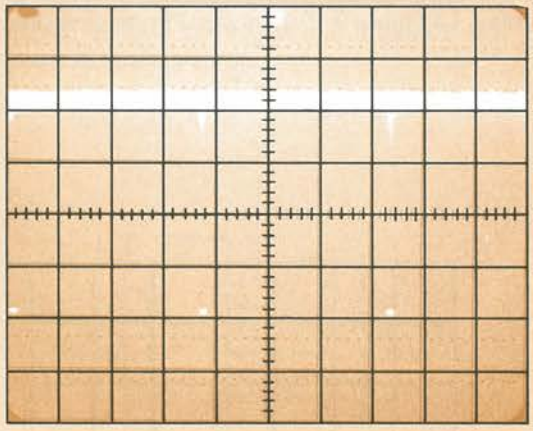


Figura 9-12. Segnale di selezione KYRD mentre sta girando il programma monitor

K) Regolate la velocità di scansione a 2 ms/div. Verificate che l'immagine sullo schermo sia simile a quella della Figura 9-13. Gli otto impulsi non sono più distinguibili singolarmente ma tutti insieme sembrano un impulso solo. Notate che vengono generati con una frequenza abbastanza bassa: il programma monitor in effetti legge la tastiera e poi passa a rinfrescare il display (operazione ben più lunga).

L) Esaminate il segnale di selezione $\overline{\text{SCAN}}$ (IC7-10). Osservatelo per differenti velocità di scansione. In Figura 9-14 è riportata l'immagine ottenuta a 200 μs/div. Questo segnale è basso ogni volta che viene letta la tastiera o viene inviato un dato al display.

SELEZIONE KYRD
(IC7-12)

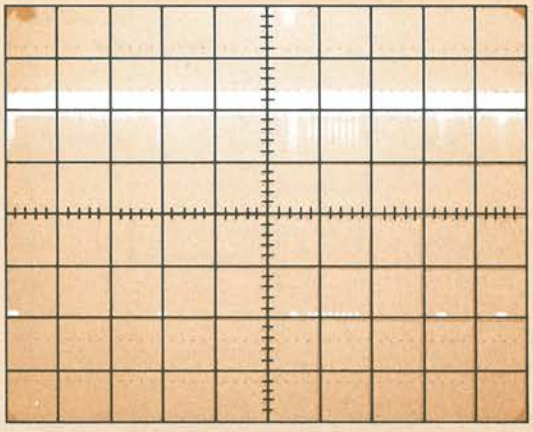


1V/div

2 ms/div

Figura 9-13. Segnale di selezione KYRD mentre sta girando il programma monitor

SELEZIONE SCAN
(IC7-10)



1V/div

200 μs/div

Figura 9-14. Segnale di selezione SCAN mentre sta girando il programma monitor

ESPERIMENTO 9-1

(continuazione)

M) Esaminate il segnale di selezione $\overline{DSP}$ (Dispaly, IC7-7). Tale segnale è basso ogni volta che viene inviato un dato al display. La Figura 9-15 vi mostra l'immagine presente sullo schermo per una velocità di 200 μs/div.

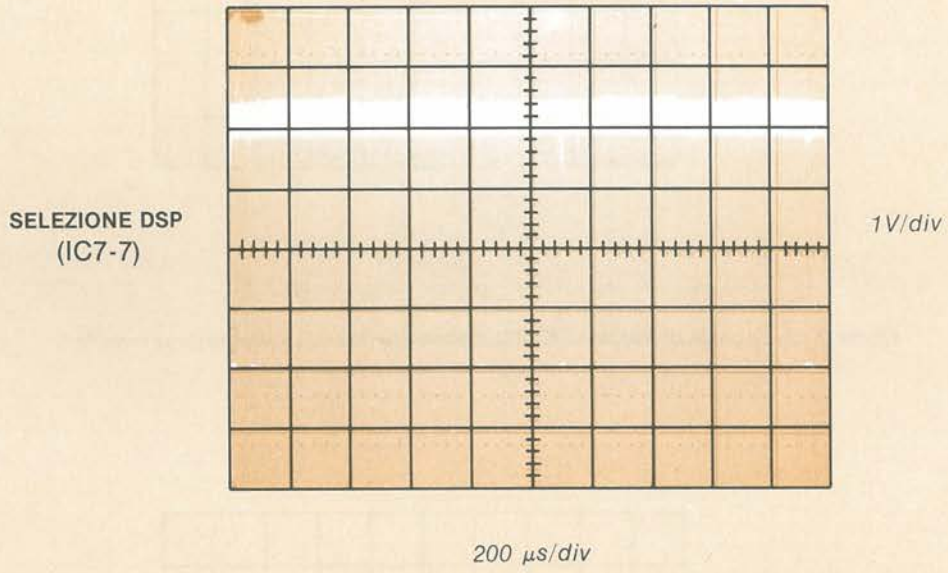


Figura 9-15. Segnale di selezione DSP mentre sta girando il programma monitor

RIASSUNTO

Sono state esaminate le forme d'onda dei segnali di selezione dei dispositivi, dapprima mentre era in esecuzione un loop breve, poi mentre stava girando il programma, abbastanza lungo, di monitor. Il programma breve vi ha permesso di esaminare le forme d'onda e di analizzare le temporizzazioni. Durante l'esecuzione del programma monitor può essere invece assai difficile osservare tali segnali. Alcune operazioni (p.es., leggere la tastiera) richiedono solo un microsecondo, all'incirca, ma vengono effettuate solo una volta ogni dieci millisecondi. Forme d'onda assai tipiche, per i segnali di selezione dispositivi, sono appunto quelle costituite da impulsi brevi e molto spaziati l'uno dall'altro.

CHIP D'INTERFACCIA A PERIFERICHE

Tutte le funzioni d'ingresso/uscita (I/O) del μLab sono effettuate utilizzando dei componenti TTL standard. In altri sistemi per semplificare parecchie operazioni connesse con l'interfacciamento, per ridurre il numero di componenti e per diminuire il costo del sistema, sono utilizzati numerosi tipi di circuiti integrati di I/O a lunga scala d'integrazione (LSI).

Di questi circuiti integrati LSI uno dei più semplici è quello detto *interfaccia periferica parallela* (Parallel Peripheral Interface) PPI, detto anche PIO (Parallel Input Output, ingresso uscita parallela). Un componente di questo tipo può essere un circuito integrato a 40 piedini, in cui sono contenute tre porte di I/O (Figura 9-16). Ogni porta può essere utilizzata come porta d'uscita o come porta d'ingresso, la direzione essendo definita da un registro di controllo presente sul chip. Un programma d'inizializzazione, contenuto nella ROM del sistema, definirà tale registro in modo da selezionare la combinazione di porte d'ingresso e di uscita voluta.

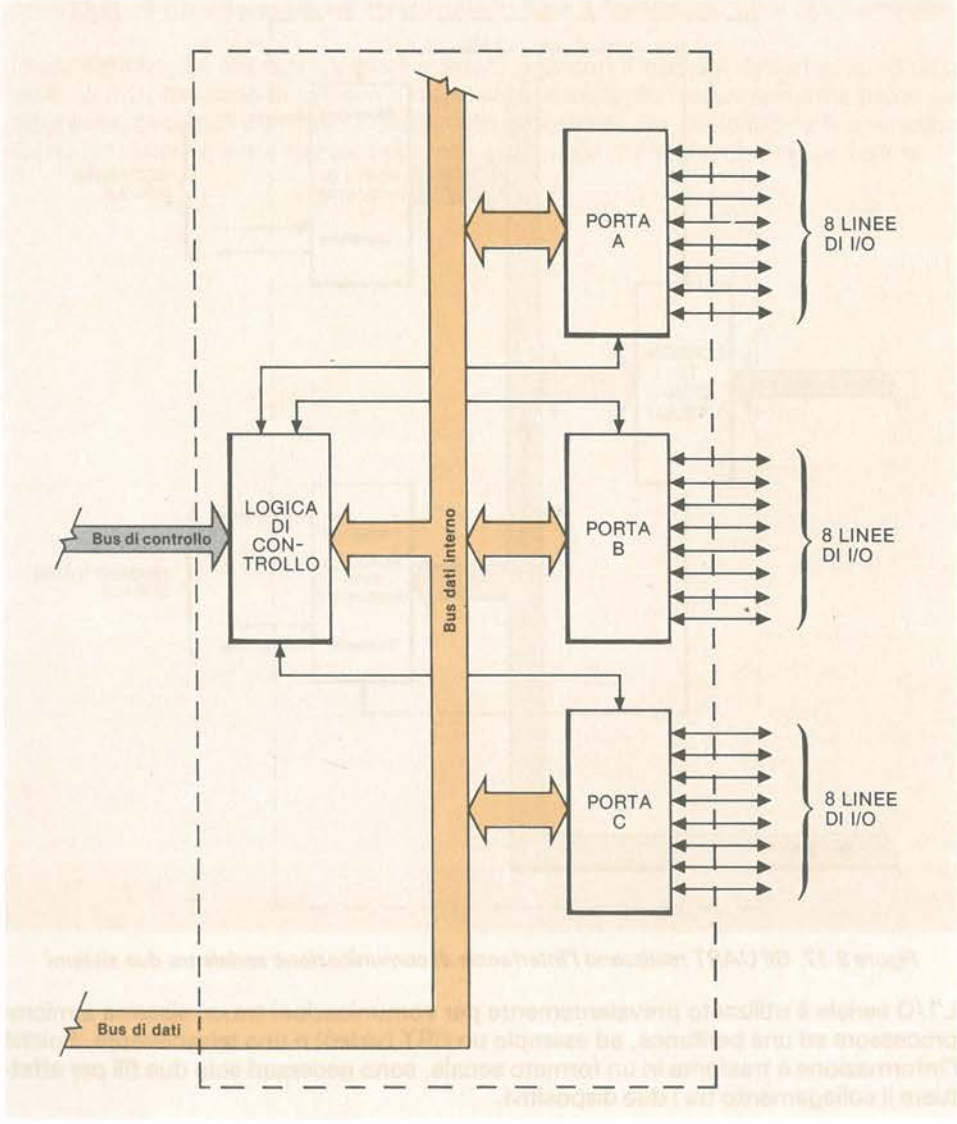


Figura 9-16. Dispositivo d'interfaccia periferiche contenente tre porte di I/O da otto bit

Questi chip hanno il vantaggio di contenere più porte nello stesso contenitore; sono inoltre assai flessibili, visto che la direzione delle diverse porte è definibile da software. Ancora, contengono in genere della logica di controllo per sincronizzare i trasferimenti e per controllare le interruzioni.

Un altro tipo di chip di interfaccia assai diffuso è quello che permette delle uscite e degli ingressi seriali (Figura 9-17); tali chip sono di solito indicati con il nome di UART (Universal Asynchronous Receiver and Transmitter, ricevitore e trasmettitore asincrono universale). Tali dispositivi sono in grado di accettare un byte di dato dal microprocessore, presentandolo a loro volta in uscita in modo seriale, un bit cioè alla volta. Il loro modo di operare è molto simile a quello dei registri a scorrimento con ingresso parallelo e uscita seriale (Parallel - In/Serial - Out Shift Register). Inoltre, possono inserire automaticamente dei bit di start, di stop, di sincronizzazione e di controllo. Il formato è definito attraverso un registro di controllo simile a quello descritto per il PPI. Gli UART possono inoltre trattare dati anche nella direzione opposta, possono cioè convertire in forma parallela, direttamente utilizzabile dal microprocessore, delle sequenze seriali di bit. Questi dispositivi sono anche detti SIO (Serial Input Output, ingresso uscita seriale), o ACIA (Asynchronous Communications Interfaced Adapter, adattatore d'interfaccia per comunicazioni asincrone).

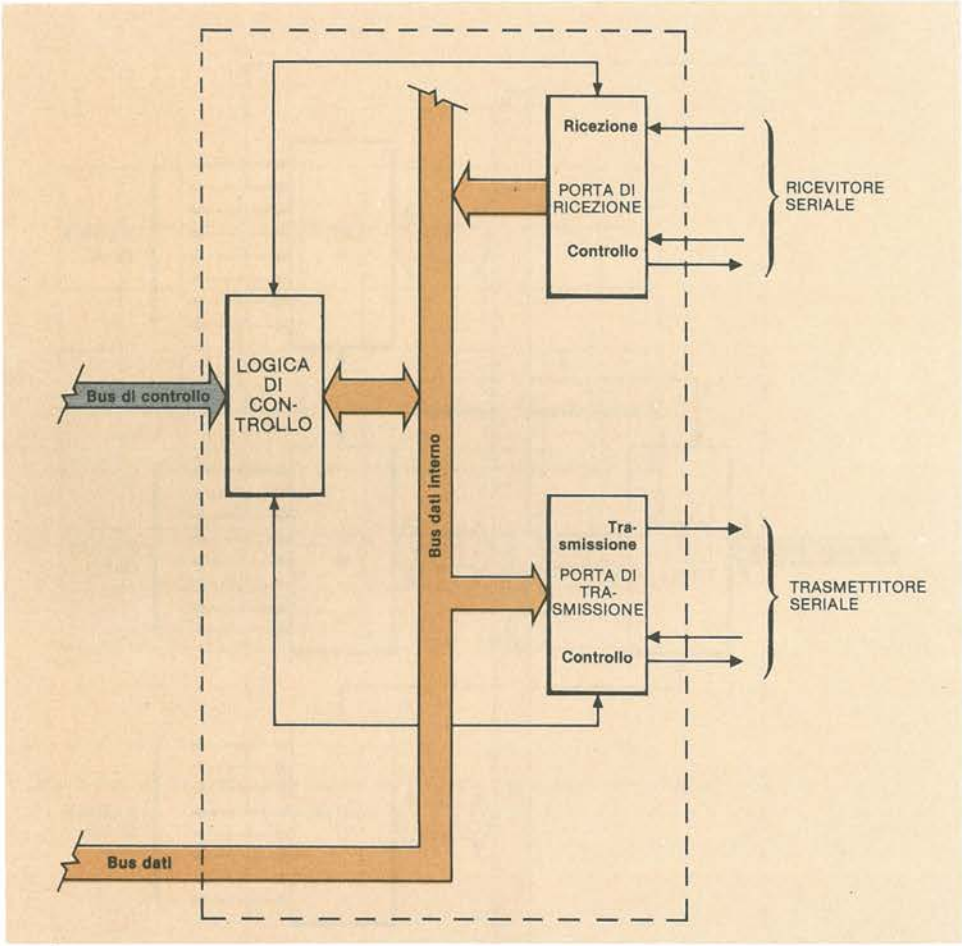


Figura 9-17. Gli UART realizzano l'interfaccia di comunicazione seriale tra due sistemi

L'I/O seriale è utilizzato prevalentemente per comunicazioni tra un sistema a microprocessore ed una periferica, ad esempio un CRT (video) o una telescrivente. Poiché l'informazione è trasferita in un formato seriale, sono necessari solo due fili per effettuare il collegamento tra i due dispositivi.

Sono disponibili numerosi altri tipi di chip specializzati di interfaccia: controllori per dischetti (Floppy Disc Controller), controllori per terminali a raggi catodici (CRT Display Controller), controllori per accesso diretto in memoria (Direct Memory Access Controller) e controllori per tastiere e display (Keyboard and Display Controller). Numerosi di questi circuiti LSI sono perfino più complessi di un microprocessore: in effetti alcuni di questi contengono, internamente al chip, un microprocessore specializzato.

Lezione 9

Le memorie a semiconduttore si suddividono in due classi principali: RAM e ROM. Nelle RAM i dati possono essere sia letti che scritti. Le RAM dinamiche perdono i dati se non sono continuamente rinfrescate, mentre le RAM statiche non hanno bisogno di rinfresco. Le ROM possono solo presentare in uscita i dati che vi sono stati programmati e possono essere classificate in base a come sono state programmate.

Le periferiche del microprocessore comunicano con il bus dei dati attraverso delle porte di I/O. Nel caso di tastiere e display, la tecnica più frequentemente usata per diminuire i circuiti di controllo è quella della scansione. Per semplificare le operazioni di I/O del sistema, sono spesso utilizzati dei chip LSI di interfaccia alle periferiche.

DOMANDE

Lezione 9

1. Il vantaggio principale delle RAM dinamiche, rispetto a quelle statiche, è che sono:
   a. non volatili.
   b. più facili da utilizzare.
   c. meno costose.
   d. tutto quanto detto nei punti precedenti.

2. Le RAM dinamiche sono più adatte per:
   a. sistemi lenti.
   b. sistemi piccoli.
   c. sistemi grossi.
   d. sistemi ad un bit.

3. Nel caso di un prodotto basato su microprocessore e realizzato in grosse quantità, la ROM che contiene il programma sarà probabilmente:
   a. ROM mascherata.
   b. PROM.
   c. EPROM.
   d. EAROM.

4. Tutti i sistemi a microprocessore hanno bisogno di periferiche per:
   a. interagire con il mondo esterno.
   b. fornire dati al microprocessore.
   c. ricevere dati dal microprocessore.
   d. tutti i punti a, b, c.

5. Il principale vantaggio della scansione è che:
   a. rende più veloci le operazioni di I/O.
   b. riduce il costo del software.
   c. riduce il costo dell'hardware.
   d. tutti i punti a, b, c.

6) Quando è effettuata la scansione dei tasti (Figura 9-5):
   a. viene letto un tasto alla volta.
   b. viene letta una riga di tasti alla volta.
   c. viene letta una colonna di tasti alla volta.
   d. vengono letti tutti i tasti in una volta.

7. L'altoparlante può generare un «bip» quando viene seguita:
   a. l'istruzione SIM.
   b. l'istruzione OUT SOD.
   c. l'istruzione BEEP.
   d. la subroutine BEEP.

8. I chip LSI di interfaccia alle periferiche sono utilizzati perché:
   a. semplificano i problemi di interfacciamento.
   b. sono flessibili.
   c. hanno un buon rapporto prezzo/prestazioni.
   d. sono tutto quanto detto ai punti a, b, c.

LEZIONE 10

Circuiti di controllo

INTRODUZIONE

In questa lezione vengono trattati i segnali di controllo utilizzati nei sistemi basati su microprocessori ed i circuiti che li generano, li trasmettono o ad essi rispondono. Su tali circuiti verranno anche fatte alcune considerazioni elettriche. Il μLab sarà il sistema di riferimento per la discussione di tali argomenti.

LA PORTA DI CONTROLLO

Sul μLab è presente una porta particolare, detta porta di controllo. Dato che la sua utilizzazione non è di evidenza immediata, può darsi che non l'abbiate ancora notata. Il microprocessore fa uso di tale porta per inviare dei segnali e dei circuiti particolari. Il bit $\overline{PROT}$, di questa porta, controlla il circuito di protezione della memoria che abbiamo già descritto in precedenza. Se tale bit è posto a uno, i primi tre quarti della RAM sono protetti contro operazioni di scrittura (Write Protected). Gli altri due bit controllano invece i circuiti, che vi permettono l'esecuzione passo passo (Single Step), legati ai modi hardware step (linea HDWR) e instruction step (linea INSTR). Di tali circuiti vi daremo una descrizione nel seguito di questa lezione.

La Figura 10-1 vi presenta il registro a 4 bit che costituisce la porta di controllo. I dati vengono immagazzinati (ingresso di CLK) dal segnale di selezione porta di controllo, che è generato dal decoder degli indirizzi. Tale situazione è del tutto simile a quella delle altre porte d'uscita. Il fatto insolito è che, nella porta di controllo, gli ingressi dei dati sono collegati al bus degli indirizzi anzichè al bus dei dati. Per questo motivo, i dati scritti in questa porta sono indipendenti dallo stato del bus dei dati.

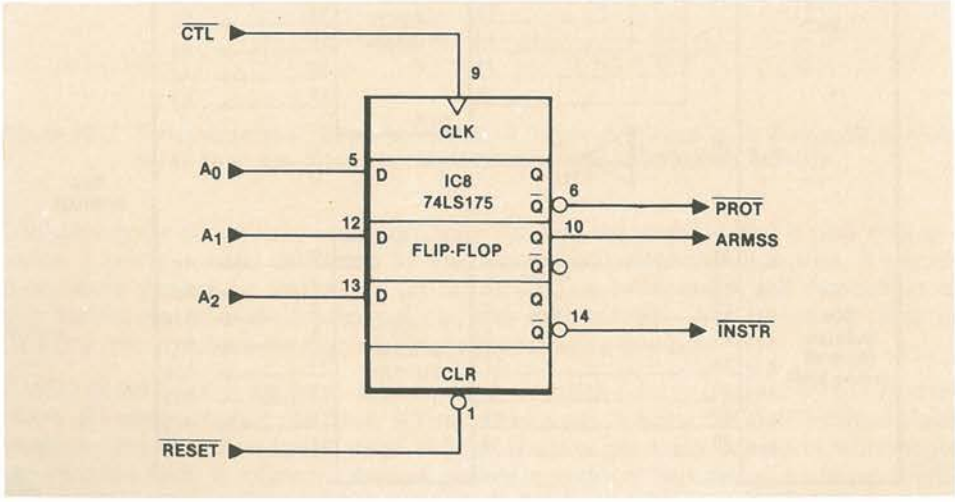


Figura 10-1. Porta di controllo del μLab

Come opera quindi tale porta? Se tornate indietro di qualche pagina ed osservate la Figura 8-1, potete vedere che la porta di controllo è selezionata da uno qualunque degli indirizzi compresi tra 1000 e 17FF. La porta potrà così essere selezionata in modo del tutto indipendente dal valore che sarà presente sulle undici linee di ordine basso degli indirizzi. Notate che gli ingressi al registro sono dati dalle linee A0, A1 e A2. È perciò l'indirizzo utilizzato che definisce il dato che deve essere scritto sulla porta. Così, ad esempio, un'operazione di scrittura all'indirizzo 1000 avrà l'effetto di azzerare tutti i bit; una scrittura all'indirizzo 1001 porterà ad uno il bit «PROT», mentre una scrittura all'indirizzo 1004 porterà ad uno il bit «INSTR».

Questa tecnica semplifica il software di controllo. Poiché non interessa quale dato venga inviato alla porta (solo gli indirizzi sono significativi), non è necessario che il software, prima di scrivere nella porta, si prepari un certo dato. L'hardware del resto non risulta più complicato di quanto non lo sarebbe se si avesse la configurazione solita.

Notate che la porta di controllo è utilizzata per alcune funzioni particolari come il single-step e la protezione della memoria. Per un sistema a microprocessore, non si tratta quindi di circuiti di controllo fondamentali.

IL BUS IN MULTIPLEX

Nel corso di questo libro ci si è sempre riferiti a due bus distinti, uno a otto bit per i dati e uno a sedici per gli indirizzi. Il microprocessore 8085, tuttavia, ha il bus dei dati che è in comune, spartito (in multiplex) con la parte inferiore del bus degli indirizzi. Gli otto bit rimanenti degli indirizzi (la parte superiore) sono invece presentati su dei piedini di indirizzo separati. Tale tecnica riduce, dal punto di vista del microprocessore, il numero di piedini necessari.

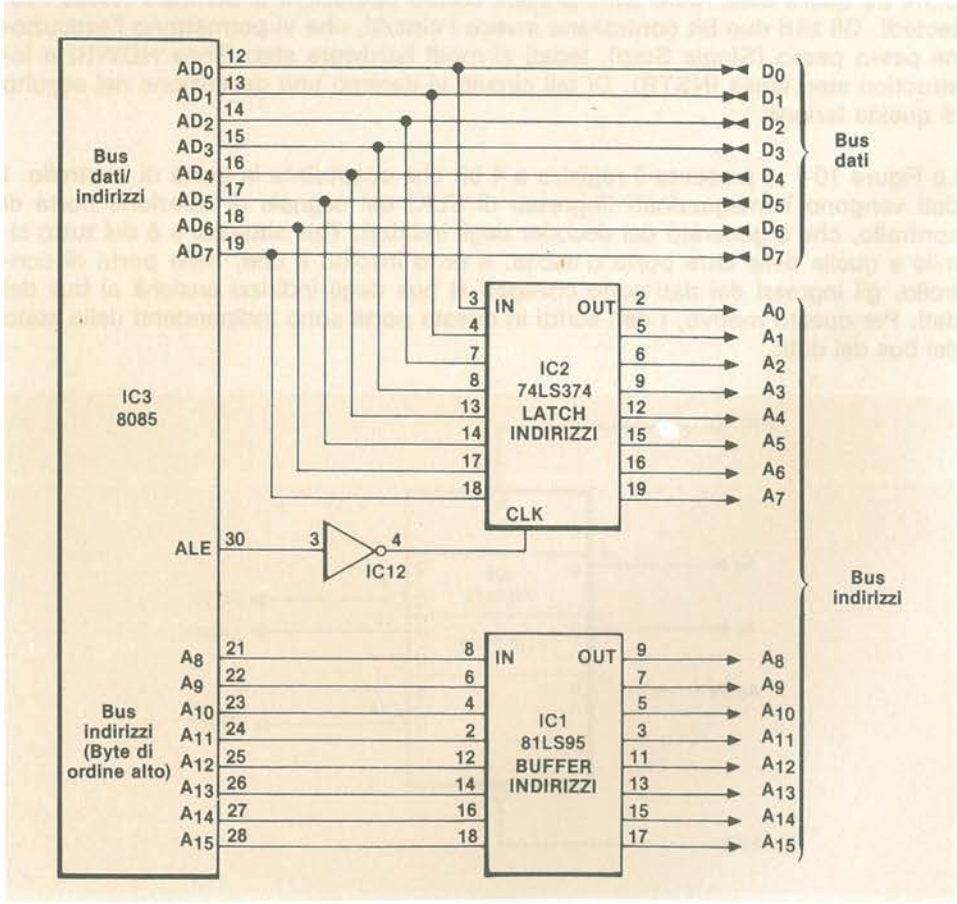


Figura 10-2. Circuito di demultiplex degli indirizzi utilizzato nel μLab

Il segnale di *abilitazione del latch degli indirizzi* (ALE, Address Latch Enable) serve ad indicare quando il bus dei dati/indirizzi contiene un indirizzo. Tale segnale è utilizzato per memorizzare il contenuto di tale bus, in modo da ottenere che anche la parte inferiore del bus degli indirizzi sia stabile (Figura 10-2).

Il dispositivo IC2 è un latch ad otto bit con uscite a tre stati. Tale registro memorizza l'informazione di indirizzo presente sul bus dei dati/indirizzi quando si verifica il fronte di discesa della linea ALE (per selezionare tale fronte è necessario l'inverter IC12). IC1 è semplicemente un buffer a tre stati che non partecipa di fatto a tale operazione (detta di demultiplex).

La Figura 10-3 vi presenta un diagramma dei tempi generale. Le linee A8-A15 contengono sempre il byte di ordine alto degli indirizzi. All'inizio di ogni ciclo di memoria, il byte di ordine basso degli indirizzi è posto sul bus dei dati/indirizzi. Il secondo fronte (quello di discesa, da alto a basso) di ALE indica che su tale bus è presente un indirizzo e permette al latch IC2 di memorizzare il byte di ordine basso dell'indirizzo.

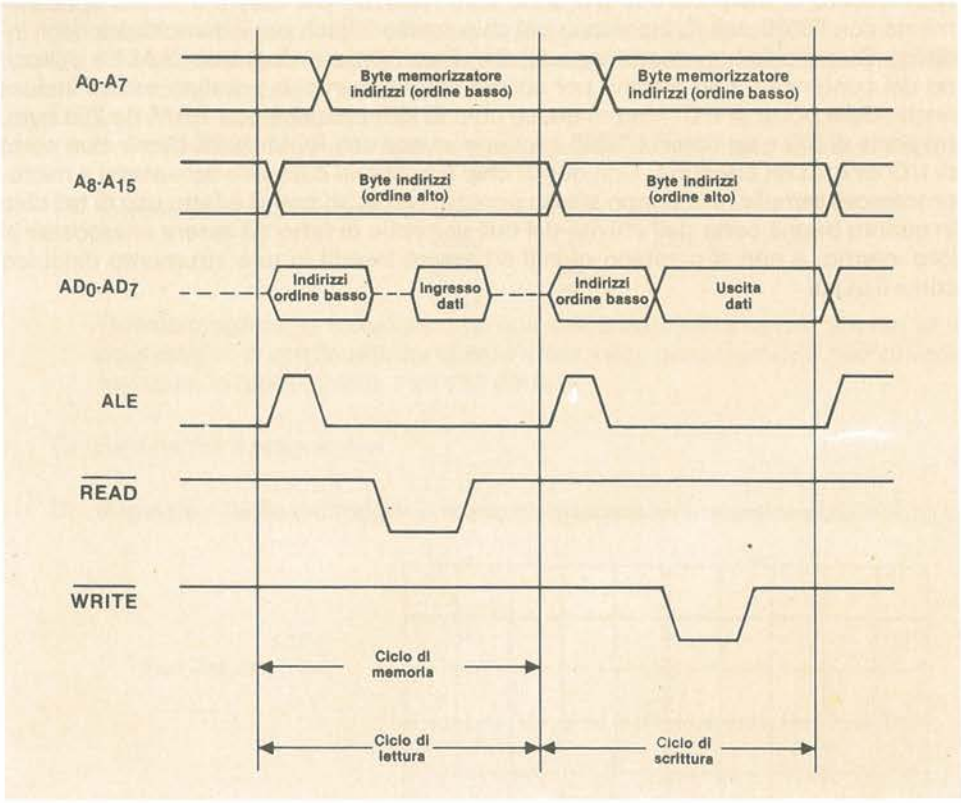


Figura 10-3. Temporizzazioni del sistema 8085. Il fronte di discesa di ALE segnala al resto del sistema che il bus dei dati/indirizzi contiene un nuovo indirizzo

L'informazione di indirizzo viene poi tolta dal bus dei dati/indirizzi e può così avvenire il trasferimento del dato. Se è in corso un'operazione di lettura, il microprocessore genera un segnale di lettura ($\overline{READ}$) e la memoria o il dispositivo di I/O indirizzato porterà il dato sul bus dei dati/indirizzi. Sul fronte di salita di $\overline{READ}$ il microprocessore leggerà il dato presente su tale bus.

Il ciclo di scrittura è del tutto simile, unica diversità il fatto che è invertita la direzione del trasferimento del dato. All'inizio del ciclo, sul bus dei dati/indirizzi viene posto il byte di ordine basso degli indirizzi e viene generato corrispondentemente un impulso ALE. Il microprocessore genera quindi un impulso di scrittura ($\overline{WRITE}$) e pone il dato sul bus dei dati/indirizzi. Sul fronte di salita di $\overline{WRITE}$, il dispositivo di memoria indirizzato memorizza il dato presente sul bus.

Grazie all'aggiunta di un latch di demultiplex (Figura 10-2), il modo di operare dei bus è così ricondotto a quello convenzionale con i bus separati. Il bus dei dati/indirizzi ritorna così ad operare semplicemente come bus dei dati. Notate che questo «bus dei dati» conterrà di fatto, all'inizio di ogni ciclo di memoria, una informazione di indirizzo. Tuttavia poiché in tale istante il bus dei dati non viene utilizzato (non sono veri né il segnale di $\overline{\text{READ}}$ né quello di $\overline{\text{WRITE}}$) non si verificherà alcun conflitto.

LA FAMIGLIA 8085

Avendo scelto il bus in multiplex, risultano liberi per altre funzioni 7 piedini del microprocessore a 40 piedini (16 linee di indirizzo più otto linee di dati sono state sostituite da 8 indirizzi, 8 dati/indirizzi più ALE). Dispositivi di memoria e di I/O standard possono del resto essere interfacciati al bus utilizzando semplicemente un latch a otto bit di demultiplex.

Numerosi IC di memoria e di I/O sono stati realizzati per essere utilizzati appositamente con l'8085: tali IC includono nel chip stesso il latch per il demultiplex degli indirizzi. Questi chip hanno otto ingressi di indirizzi/dati e un ingresso di ALE e utilizzano dei contenitori a 40 piedini, per cui nel chip di memoria possono essere incluse anche delle porte di I/O. Uno di questi chip (8155) contiene una RAM da 256 byte, tre porte di I/O e un timer. L'8355 contiene invece una ROM da 2K byte e due porte di I/O da otto bit ciascuna. Con questi chip è possibile costruire dei sistemi a microprocessore semplici e al tempo stesso potenti. Nel μLab non si è fatto uso di tali chip in quanto buona parte dell'attività del bus verrebbe di fatto ad essere «nascosta» al loro interno, e non si prestano quindi ad essere inseriti in uno strumento didattico come il μLab.

Temporizzazioni del bus

INTRODUZIONE

In questo esperimento, facendo uso di un oscilloscopio, osserverete il modo di operare del bus dei dati/indirizzi e potrete così verificare direttamente il diagramma dei tempi riportato nella Figura 10-3.

PROCEDIMENTO

A) Preparate l'oscilloscopio effettuando i seguenti collegamenti:
1. Il canale A sulla linea D0 del bus dei dati (inserite il puntale della sonda nel foro metallizzato D0).
2. Il canale B sulla linea ALE (IC12-3).
3. Sincronizzatevi sul canale A.
4. Per entrambi i canali impostate una scala di 2 V/div e una velocità di scansione di 1 μs/div.

B) Introducete, tramite tastiera, il seguente programma:

0800	C3	LOOP:	JMP	LOOP
0801	00			
0802	08			

Questo programma è costituito da una sola istruzione di salto che salta su se stessa. Il microprocessore esegue in continuazione questo breve loop, permettendovi così di vedere visualizzata sull'oscilloscopio, in modo chiaro, l'attività dei bus.

C) Fate partire il programma.

D) Regolate il livello del trigger in modo da ottenere un'immagine stabile sullo schermo. (Figura 10-4).

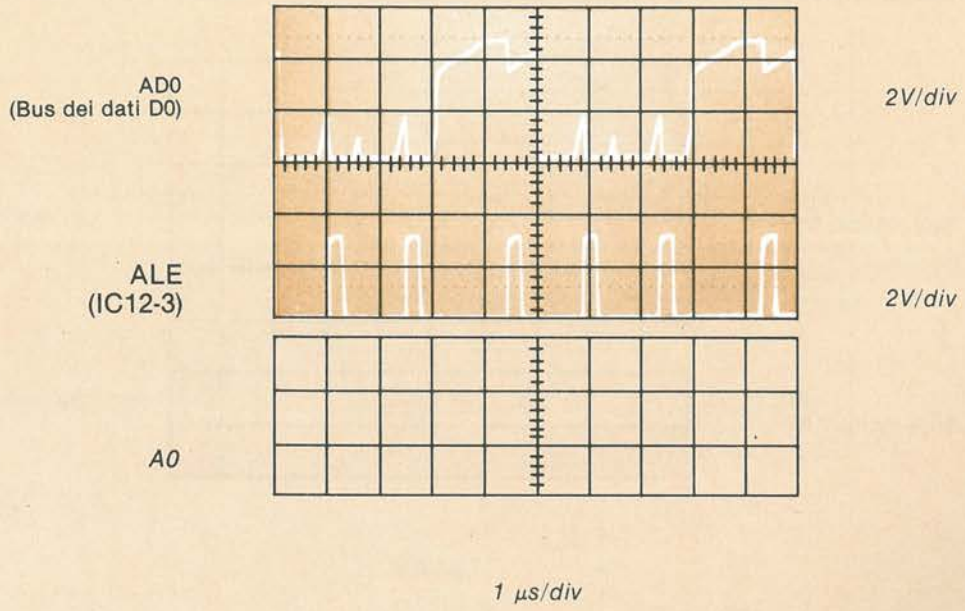


Figura 10-4. I fronti di discesa di ALE indicano che su AD0 sono presenti degli indirizzi stabili

(continuazione)

Il canale A è collegato alla linea del bit 0 dei dati/indirizzi (AD0). (Se questo non vi fosse del tutto chiaro ritornate alla Figura 10-2). Questa forma d'onda è un po' confusa: tra i livelli, ben definiti, di zero e di uno si vedono degli altri livelli strani. Questa situazione è dovuta al fatto che la linea è in qualche momento aperta, fluttuante (cioè non è pilotata da alcun dispositivo).

E) Ricordate che il segnale ALE indica che sul bus dei dati/indirizzi è presente un indirizzo. Con il fronte di discesa di ALE, lo stato della AD0 proveniente dal microprocessore è memorizzato nel latch degli indirizzi, in modo da generare il segnale A0. Notate che quando si verifica tale fronte, la linea AD0 contiene sempre un segnale stabile e di livello valido.

F) Disegnate su di un foglio, riferendovi alla Figura 10-4, la forma d'onda dei segnali d'indirizzo. Tale forma d'onda ha delle transizioni solo in corrispondenza dei fronti di discesa di ALE, quando viene ad assumere il valore di AD0.

G) Collegate il canale B alla linea A0 (inserite il puntale della sonda nel foro metallizzato). Verificate che l'immagine sullo schermo sia simile a quella della Figura 10-5. La forma d'onda di A0 dovrebbe essere simile a quella che voi avete disegnato basandovi sulla Figura 10-4.

H) Collegate il canale A ad A0 e il canale B ad ALE. La Figura 10-6 vi mostra l'immagine che dovreste osservare sullo schermo. Potete vedere che la linea A0 viene modificata solo quando si verifica il fronte di discesa di ALE.

I) Collegate il canale A ancora ad AD0 e il canale B alla linea di $\overline{READ}$ (servitevi del punto di test $\overline{READ}$ per l'analisi della firma, posto subito sotto la fila dei LED del bus degli indirizzi). La Figura 10-7 vi mostra l'immagine che vi si presenterà sullo schermo. Quando $\overline{READ}$ è basso la memoria pone un dato sul bus dei dati/indirizzi. Sul fronte di salita di $\overline{READ}$ il microprocessore legge tale dato. Notate che in tale istante la linea AD0 è ad un livello stabile e valido.

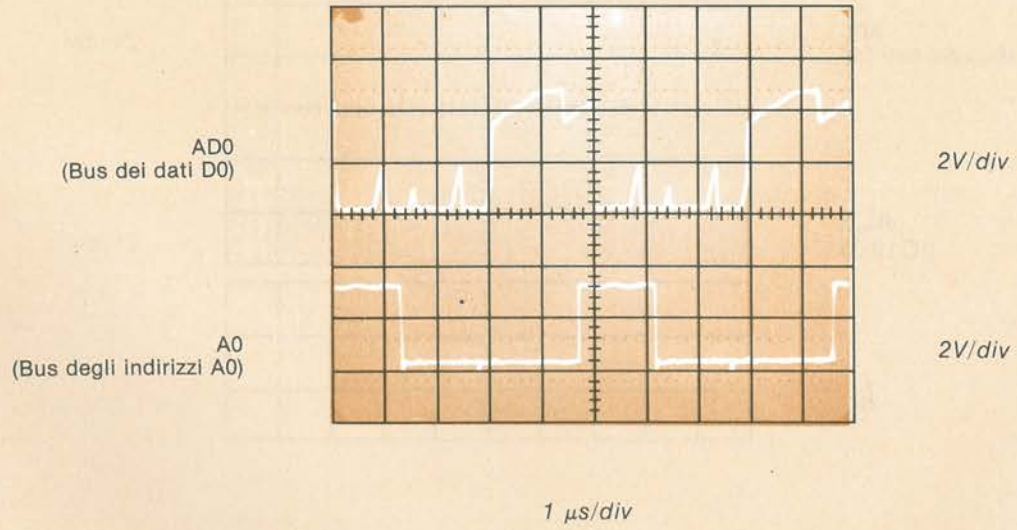


Figura 10-5. La linea di indirizzo A0 dopo il demultiplex, generata a partire dalla linea di indirizzo in multiplex AD0

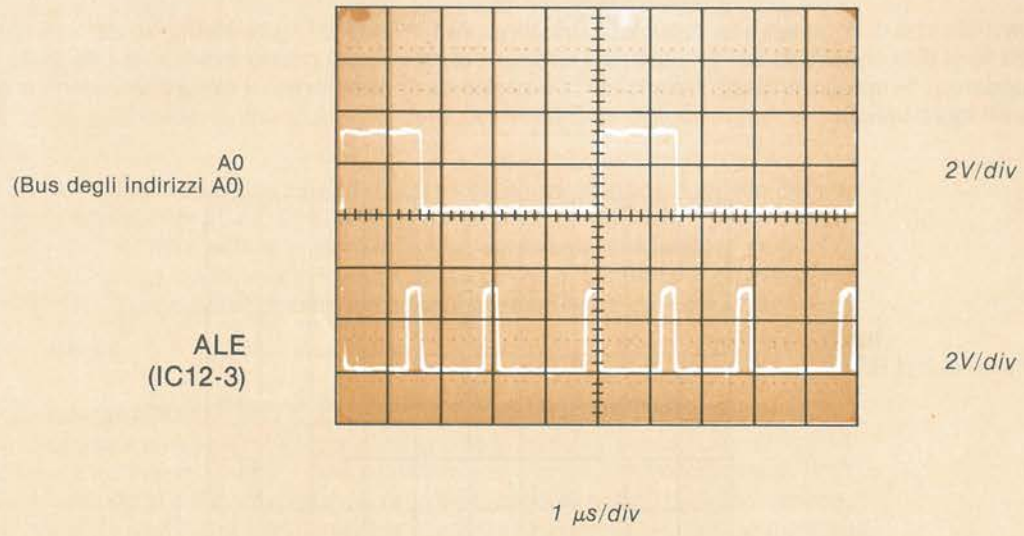


Figura 10-6. Il segnale ALE controlla A0

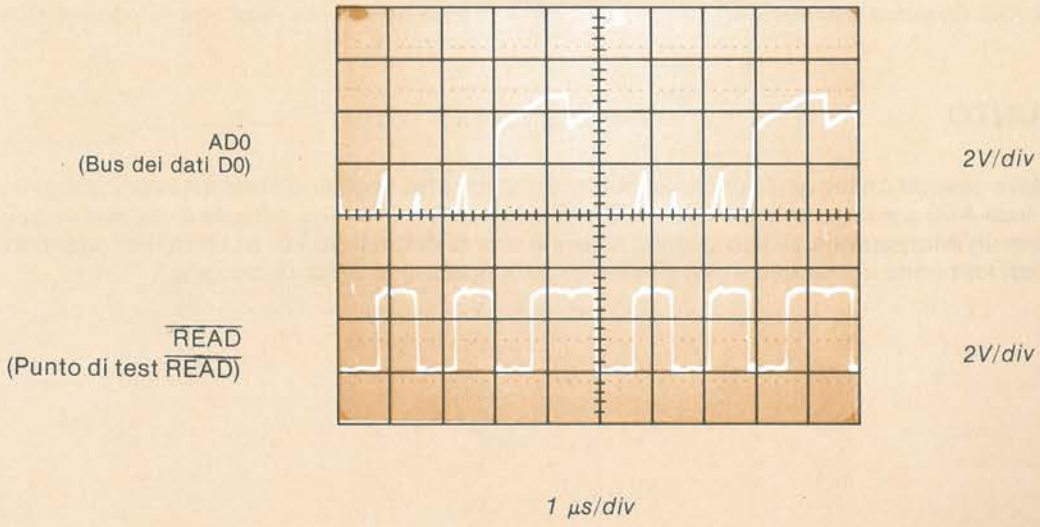


Figura 10-7. I dati stabili presenti sulla linea AD0 sono letti dal microprocessore sul fronte di salita del segnale READ

(continuazione)

J) Osservate che ci sono dei casi in cui $\overline{READ}$ è alto e ALE è basso (Figura 10-8). Quando questo si verifica vuol dire che il bus dei dati/indirizzi non è utilizzato e può perciò trovarsi in uno stato ad alta impedenza. Si spiega in questo modo perché alcune parti della forma d'onda di AD0 non si trovano a livelli logici validi.

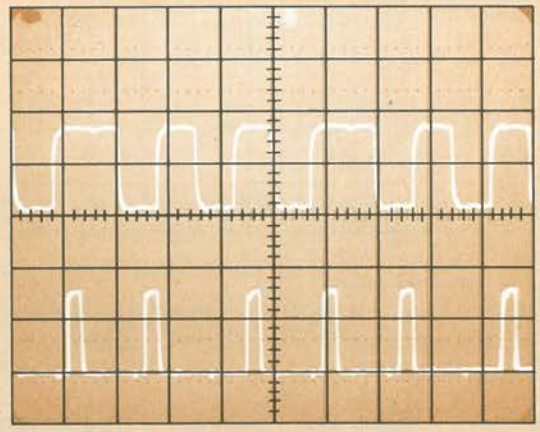


Figura 10-8. Quando il segnale $\overline{READ}$ è alto e ALE è basso, la linea AD0 non contiene segnali logici significativi

RIASSUNTO

Mentre stava girando un breve programma, abbiamo esaminato i segnali del bus dei dati/indirizzi (in multiplex). La linea AD0 contiene un'informazione d'indirizzo quando si verifica il fronte di discesa di ALE, mentre contiene un'informazione di dato quando si ha il fronte di salita di $\overline{READ}$. In alcuni altri istanti il bus dei dati/indirizzi non viene utilizzato e si può trovare quindi a qualunque livello di tensione.

Clock

Tutti i microprocessori sono dei circuiti digitali sincroni ed hanno perciò bisogno di un clock (orologio). La maggior parte dei primi tipi di microprocessori avevano bisogno di circuiti esterni che generassero, per il clock, delle forme d'onda adatte. Tensioni, fasi e temporizzazioni relative a tale linea hanno spesso delle specifiche critiche. I produttori di questi processori (p.es. 6800, 8080) forniscono, per generare il clock necessario al processore, dei circuiti integrati di clock e di temporizzazione appositamente progettati. Di solito tali circuiti sono controllati da un cristallo di quarzo.

I processori più recenti hanno invece dei circuiti di clock interni, grazie ai quali, per generare il clock richiesto, basta collegare il cristallo direttamente a due piedini del microprocessore (è questo il caso dell'8085 utilizzato nel μLab). Alternativamente quando è importante il fattore costo e quando non è richiesta alcuna precisione, il cristallo è spesso sostituito da un circuito di temporizzazione RC. Parecchi microprocessori possono inoltre essere sincronizzati da un segnale TTL di clock, generato da un clock principale di sistema, in modo tale che è possibile la sincronizzazione di più processori.

In generale il processore opera ad una frequenza che è una frazione di quella del cristallo. L'8085 utilizzato nel μLab, ad esempio, ha un cristallo da 4 MHz. Il ciclo macchina tuttavia è di 2 MHz, e il segnale Clock Out (generato dall'8085) è pure di 2 MHz. La maggior parte dei microprocessori MOS opera a frequenze comprese tra 100 KHz a 10 MHz. I processori bipolari bit-slice possono invece operare a frequenze più alte.

Reset

Il piedino di *Reset* dell'8085 è utilizzato per inizializzare il sistema all'accensione (Figura 10-9). Quando a tale piedino è applicato un livello basso, i circuiti all'interno del microprocessore vengono azzerati, cioè inizializzati. Il contatore di programma (PC) viene caricato con 0000 e l'esecuzione del programma parte perciò da tale indirizzo, dove inizia infatti la routine di inizializzazione contenuta nella ROM. Su un qualunque sistema (e così anche nel μLab), la procedura di inizializzazione all'accensione fa

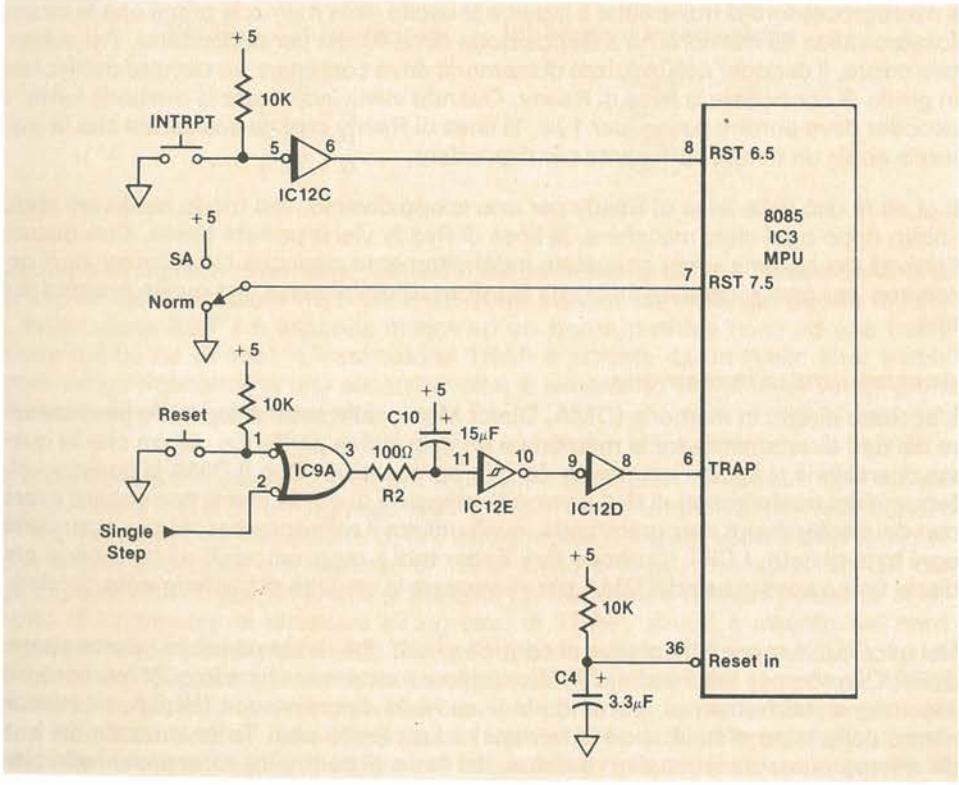


Figura 10-9. Circuiteria d'interruzione del μLab

sì che vengano eseguiti alcuni test di verifica del sistema e successivamente porta le periferiche nelle condizioni iniziali volute.

Per richiamare alta la linea è utilizzata una resistenza, mentre un condensatore collegato a massa serve a generare automaticamente un impulso di reset all'accensione. Quando viene data tensione, il condensatore è scarico e si trova ad un livello logico basso. Tale livello è perciò presente sull'ingresso di Reset In. Fintanto che rimane basso, il processore rimane nello stato di reset. Quando, grazie alla resistenza di richiamo, il condensatore è caricato ad una tensione superiore al livello di soglia dell'ingresso (su tale linea, internamente al chip, è posto uno Schmitt trigger che serve ad eliminare i disturbi durante le transizioni), il processore inizia ad eseguire il programma partendo dall'indirizzo 0000. Notate che il pulsante RESET del μLab non è collegato all'ingresso di Reset del microprocessore. Se il pulsante di RESET fosse stato collegato direttamente a tale ingresso, ad ogni pressione di tale tasto, il μLab avrebbe effettuato la routine di azzeramento della memoria che viene eseguita nel corso dell'inizializzazione. Questo provocherebbe la cancellazione di tutti i programmi memorizzati nella RAM. Il tasto di RESET, invece, è collegato alla linea di ingresso *Trap* di cui daremo più oltre la descrizione.

Stato

L'8085 ha due particolari uscite di stato (Status), S0 e S1, che danno delle ulteriori informazioni riguardo al ciclo macchina corrente. Tali linee indicano inoltre se il processore si trova nello stato di HALT. Il μLab tuttavia non fa uso di questi segnali in quanto vengono utilizzati solo in alcune applicazioni particolari.

Ready

Quando l'ingresso di Ready (pronto) va basso, il microprocessore entra in uno stato di *wait* (attesa). In tale stato, i bus mantengono il loro stato logico corrente finché la linea Ready non ritorna alta. Lo stato di wait fa sì che il microprocessore possa utilizzare dei dispositivi di memoria o di I/O lenti. Così per esempio se un chip di memoria richiede che l'indirizzo sia stabile per 1 μs prima di poter dare dei dati validi in uscita, il microprocessore si troverebbe a leggere le uscite della memoria prima che le stesse fossero valide (la memoria ha a disposizione circa 400 ns per rispondere). Per evitare tale errore, il decoder dell'indirizzo di memoria deve contenere un circuito particolare in grado di controllare la linea di Ready. Quando viene indirizzata la memoria lenta, il decoder deve portare bassa, per 1 μs, la linea di Ready così da assicurare che la memoria abbia un tempo sufficiente per rispondere.

Il μLab fa uso della linea di Ready per uno scopo diverso. Nel modo hardware step, subito dopo ogni ciclo macchina, la linea di Ready viene portata bassa. Con questo l'attività del sistema viene congelata indefinitamente cosicché l'utilizzatore può osservare, per ogni ciclo macchina, sia le informazioni di stato che quelle presenti sui bus.

Accesso diretto in memoria

L'accesso diretto in memoria (DMA, Direct Memory Access) è un modo per trasferire dei dati direttamente tra la memoria e un dispositivo periferico, senza che in questa operazione si abbia l'intervento del microprocessore. Con il DMA si possono effettuare dei trasferimenti di dati a velocità maggiori di quelle che si potrebbero avere con dei trasferimenti «programmati», quali utilizza il microprocessore per scambiare ogni byte di dato. I CRT (Cathode Ray Tube, tubi a raggi catodici) e i controllori per dischi fanno spesso uso del DMA per aumentare la velocità di trasferimento dei dati.

Nel microprocessore è l'ingresso di controllo *Hold* che rende possibile questa operazione. Quando tale linea è alta, l'8085 completa il ciclo macchina in quel momento in esecuzione e si ferma poi, portando la linea *Hold Acknowledge* (HLDA, riconoscimento dello stato di hold, cioè di fermata) ad un livello alto. Tutte le uscite dei bus del microprocessore (i bus degli indirizzi, dei dati e di controllo) sono poste nello stato ad alta impedenza. Un dispositivo periferico può a tal punto prendere il controllo di tali bus ed effettuare tutti i trasferimenti di dati necessari. Quando la linea di Hold

è riportata bassa dalla periferica, il microprocessore può riprendere l'operazione dal punto in cui era stato fermato.

Le periferiche devono disporre di un controllo di DMA che generi i segnali di indirizzo e di controllo. Sono disponibili dei chip controllori di DMA che svolgono appunto tale funzione.

Notate che, se in un sistema a microcalcolatore è presente la possibilità del DMA, tutti i buffer dei bus devono essere dei bus a tre stati. La linea HLDA disabilita le uscite dei buffer e permette così al controllore del DMA di utilizzare i bus.

Interruzioni

Le interruzioni (Interrupt) fanno sì che dell'hardware esterno possa richiedere un intervento immediato da parte del microprocessore. Esse interrompono il flusso normale del programma e trasferiscono il controllo ad una particolare routine software. In questa lezione viene descritto l'hardware necessario per generare un'interruzione. Le implicazioni software di tale problema sono invece state descritte nella Lezione 6. Nella descrizione che seguirà, si assume che l'interruzione in questione sia già stata abilitata dal software.

Nell'8085 ci sono due gruppi di interruzioni. A ciascuna delle interruzioni del primo gruppo (TRAP, RST 5.5, 6.5 e 7.5) corrisponde, sul microprocessore, un piedino distinto. Il secondo gruppo (RST 1, 2, 3, 4, 5, 6 e 7) è invece controllato dalle linee INTR e $\overline{\text{INTA}}$.

Per generare una delle interruzioni del primo gruppo è sufficiente applicare un segnale al piedino corrispondente sul microprocessore. La routine di servizio dell'interruzione relativa a tale piedino è così automaticamente indirizzata e chiamata. La Tabella riporta l'indirizzo associato ad ogni piedino che è definito dal progetto stesso del microprocessore 8085.

Piedino d'interruzione	Indirizzo d'interruzione
TRAP	0024
RST 5.5	002C
RST 6.5	0034
RST 7.5	003C

Gli ingressi d'interruzione sono attivati o da un fronte o da una variazione di livello logico. Le interruzioni RST 5.5 e 6.5 sono attivate da un livello alto (1 logico). L'interruzione RST 7.5 risponde invece ad un fronte positivo (cioè ad una transizione dal basso all'alto). L'interruzione TRAP è attivata da un livello alto; perché però venga riconosciuta una seconda volta, è necessario che la linea venga prima riportata bassa e poi ancora alta.

Il μLab utilizza l'ingresso di Trap per il pulsante di RESET, l'ingresso RST 6.5 per il pulsante di INTRTP e l'ingresso RST 7.5 per l'interruttore «SA» (nella Lezione 17 è presentata una descrizione di tale interruttore). I circuiti sono molto semplici, con la sola eccezione dell'ingresso di TRAP (Figura 10-9). Un dispositivo OR con gli ingressi attivi bassi (in effetti si tratta di un dispositivo NAND) permette al circuito di single-step di accedere all'ingresso di TRAP, su cui è inserito dell'hardware per eliminare i rimbalzi (R2, C10 e l'ingresso a Schmitt trigger IC12E). La resistenza da 100 ohm e il condensatore puliscono i rimbalzi del segnale di RESET in modo da garantire che, ogni volta che viene premuto un tasto, venga generata una sola interruzione. Non è necessario invece effettuare un'operazione di antirimbalzo nel caso degli altri ingressi di interruzione, in quanto possono essere disabilitati da software appena vengono riconosciuti ed è così possibile impedire che si verifichi una seconda interruzione.

È invece più difficile utilizzare le interruzioni del secondo gruppo. L'interruzione viene iniziata dall'ingresso di *Intr* (Interrupt Request, richiesta di interruzione). Il circuito che richiede l'interruzione deve presentare sul bus dei dati un codice, che indica a quale indirizzo è posta la routine di servizio dell'interruzione che deve essere eseguita. I tempi di tale operazione sono controllati e coordinati dal piedino $\overline{Inta}$ (Interrupt Acknowledge, riconoscimento dell'interruzione) disponibile come uscita sul microprocessore. L'interruzione Intr non è utilizzata sul μLab.

Priorità delle interruzioni

È possibile, e non del resto vietato, che si verifichino più richieste d'interruzioni simultanee. Ad ogni interruzione è assegnata una certa priorità e viene riconosciuta per prima l'interruzione di priorità più elevata. TRAP ha la priorità più elevata; seguono RST 7.5, 6.5 e 5.5 in ordine decrescente. Ad INTR è assegnata la priorità più bassa.

IL CIRCUITO DI SINGLE STEP

Il μLab contiene un particolare circuito di controllo che serve a realizzare le funzioni di single-step (passo singolo). La Figura 10-10 riporta la parte del circuito di

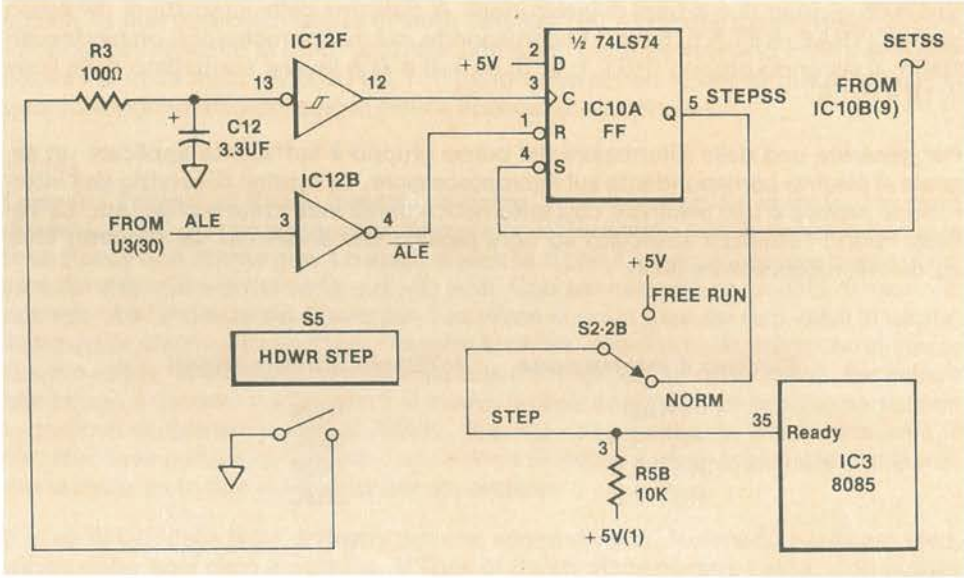


Figura 10-10. Il circuito di single-step fa sì che il microprocessore proceda di un solo ciclo macchina

single-step che serve a far procedere di un ciclo macchina il microprocessore ogni volta che viene premuto il tasto HDWR STEP.

Per tale circuito, la sequenza di eventi è la seguente:

1. Quando gira il programma monitor, il segnale di controllo SETSS (Set Single-Step, attiva il passo singolo) è basso (IC10-9), così da forzare il segnale STEPSS (Step Single-Step) ad essere alto (IC10-5). Tale linea è collegata all'ingresso Ready del microprocessore (IC3-35) attraverso l'interruttore S-2. Il livello alto sulla linea Ready permette al sistema di operare alla sua velocità normale.

2. Quando sul display appare un indirizzo valido e viene premuto il tasto HDWR STEP, il programma monitor trasferisce l'operazione al programma utente, all'indirizzo specificato, e porta quindi la linea SETSS alta.

3. Al ciclo macchina successivo, sulla linea $\overline{ALE}$ si verifica un impulso e viene così azzerato (reset) il latch IC10A e portata a sua volta bassa la linea

STEPSS. Come conseguenza la linea di Ready viene ad essere bassa.

4. Con la linea di Ready bassa, tutta l'attività del sistema si ferma. Il μLab si trova ora nel modo single-step hardware. L'indirizzo che era precedentemente visualizzato sul display è ora presente sui LED del bus degli indirizzi.

5. Quando viene premuto un'altra volta il tasto HDWR STEP, sul piedino di ingresso del clock (IC10-3) si verifica un fronte di salita e la linea STEPSS (e con essa la linea di Ready) si porta di conseguenza alta. Il tasto HDWR STEP è protetto dai rimbalzi attraverso R3, C12 e IC12F.

6. Con la linea di Ready ancora alta, il microprocessore riprende l'esecuzione normale del programma ed effettua il ciclo macchina successivo.

7. Appena viene eseguito il prossimo ciclo macchina ed è stato memorizzato nel latch un nuovo indirizzo, l'impulso sulla linea $\overline{ALE}$ azzera ancora il latch IC10A e la linea STEPSS ritorna bassa.

8. Questa sequenza viene ripetuta ogni volta che viene premuto il tasto HDWR STEP.

9. Quando viene premuto il tasto RESET, viene azzerato IC10B, la linea SETSS è riportata bassa e il segnale STEPSS è forzato di conseguenza alto. Il segnale di RESET è anche portato all'ingresso di interruzione Trap del microprocessore (IC3-6). Tale interruzione restituisce al programma monitor il controllo del sistema.

TIMER PROGRAMMABILI

Le interruzioni sono spesso utilizzate in coppia con degli interval timer (temporizzatori di intervalli) programmabili. Fondamentalmente si tratta di dispositivi contatori, che vengono fatti partire da un comando generato dal microprocessore e che contano quindi ad una frequenza nota (in genere sono controllati da un cristallo). Quando viene raggiunto il conteggio programmato, il timer genera una interruzione al microprocessore. Il microprocessore, grazie a questa interruzione generata dal contatore, può svolgere delle operazioni, che devono essere effettuate in tempi ben precisi, senza far ricorso per la generazione di tali tempi a dei loop software (come invece si verifica nel caso del μLab). Il microprocessore, mentre il tempo è in tal modo tenuto sotto controllo, può eseguire altri programmi. La maggior parte dei timer oggi disponibili possono svolgere anche altre interessanti funzioni di temporizzazione.

Nel μLab i loop software di temporizzazione, utilizzati per la routine di scansione del display, potrebbero così essere sostituiti da un circuito timer. Liberando il microprocessore da questi compiti che richiedono parecchio tempo, tra ogni scansione del display sarebbe così possibile eseguire dei programmi ben più lunghi.

CONSIDERAZIONI ELETTRONICHE

Una delle proprietà dei sistemi digitali è quella di poter essere analizzati da un punto di vista logico. Potete ragionare in termini di acceso e spento, on e off, uno e zero, sì e no. Diversamente da quanto si verifica nei circuiti analogici, sia la matematica che la teoria elettronica servono solo limitatamente. I circuiti integrati sono spesso considerati delle scatole nere. Per assicurare tuttavia una corretta operatività dei circuiti logici, è necessario tener conto di alcuni fattori elettronici.

Carico

Il carico d'uscita è uno dei fattori più importanti da considerare. L'uscita di un dispositivo può fornire solo una certa quantità di corrente e se si cerca di superare tale limite il circuito non potrà operare in modo corretto. Se il circuito è eccessivamente sovraccaricato, i livelli logici possono risultare alterati o il dispositivo può scaldarsi troppo e danneggiarsi irrimediabilmente.
In un sistema a microprocessore bisogna di solito tener conto di due tipi di carico,

quello statico e quello dinamico. I carichi statici dipendono dalle componenti resistive e di corrente presenti nei nodi logici. I carichi dinamici sono invece legati alle capacità dei nodi. Nel caso di uscite MOS che pilotano più ingressi TTL, il primo problema da considerare è quello dei carichi statici. Tuttavia, quando si hanno numerosi dispositivi presenti su di un bus e quando il bus si estende a più schede, passando anche attraverso dei connettori, diventa significativo il problema dei carichi dinamici. Mentre i carichi statici modificano di solito i livelli logici e i margini di rumore, i carichi dinamici hanno influenza sulla velocità e sulle temporizzazioni, in quanto rallentano i fronti di transizione tra i livelli.

Il problema dei carichi statici è particolarmente importante nel caso dei sistemi a microprocessore, in cui si possono avere più dispositivi collegati allo stesso bus. Le uscite degli indirizzi del microprocessore devono essere in grado di pilotare tutti i dispositivi collegati al bus degli indirizzi. Ogni dispositivo poi, che pilota il bus dei dati, deve essere a sua volta in grado di pilotare tutti i dispositivi collegati a tale bus.

Il μLab si serve di buffer per pilotare tutte le linee di indirizzo (diciamo così che il bus è «bufferato»), in quanto dispone di una corrente sufficiente ad alimentare i LED del bus degli indirizzi.

Anche su una parte del bus dei dati (Figura 10-11) sono posti dei buffer. Per poter introdurre, sul bus dei dati, dei buffer direttamente all'uscita del microprocessore, sarebbe necessario che gli stessi fossero bidirezionali, in quanto sul bus dei dati i dati vengono trasferiti in entrambe le direzioni. Il μLab invece fa uso di un buffer unidirezionale (IC14) in modo da introdurre sul bus dei dati un disaccoppiamento solo verso l'esterno e pilotare così le porte d'uscita e i LED del bus dei dati.

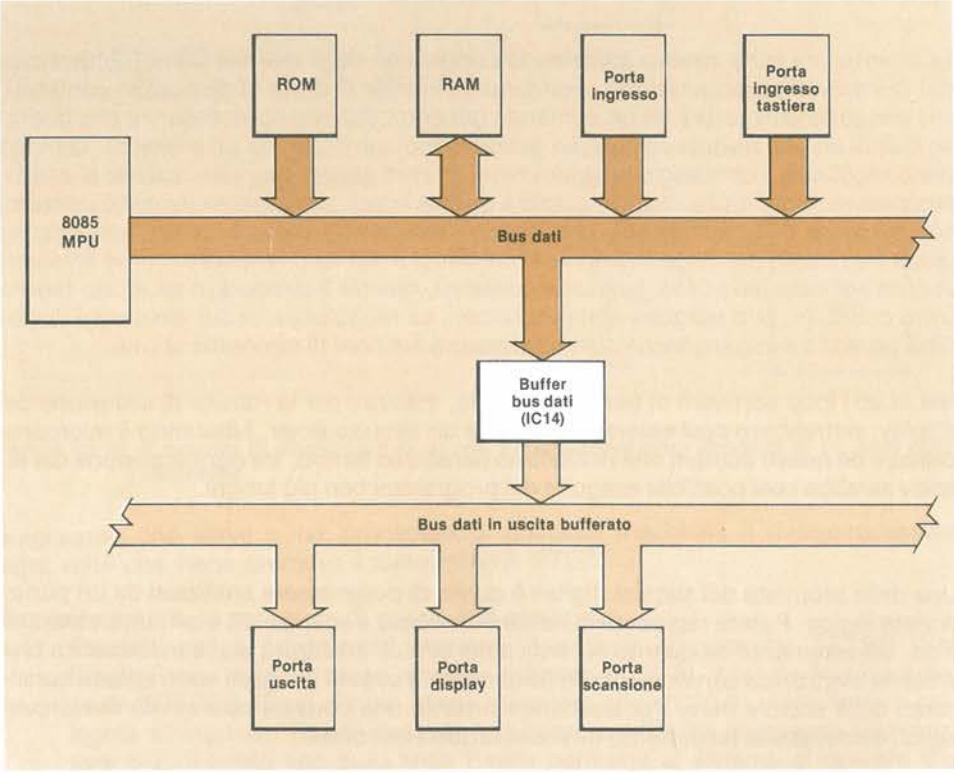


Figura 10-11. Buffer sul bus dei dati del μLab

Mentre il μLab fa uso, per il bus degli indirizzi, dei dati e di controllo, di buffer non invertenti, numerosi altri sistemi utilizzano invece dei buffer invertenti, il che comporta avere dei bus in logica negativa. Questa scelta non dà luogo ad alcun problema fintanto che avete ben presente tale fatto: è il caso ad esempio dei LED della porta d'uscita del μLab (i LED danno una indicazione in logica negativa). I buffer invertenti

sono utilizzati perché spesso sono più veloci, meno costosi e consumano meno dei buffer non invertenti.

Bypass

Come tutti i sistemi digitali, è necessario prestare attenzione perché le linee di alimentazione e di massa siano pulite e senza disturbi. Vengono a tal fine sparsi sulla scheda del circuito stampato dei condensatori di bypass, di solito da 0,1 a 0,01 μF. Per ogni alimentazione da 5 Volt inoltre viene di solito posto sulla scheda un condensatore al tantalio da 10 μF o maggiore.

Temporizzazioni

Perché un sistema a microprocessore possa operare in modo corretto, devono essere soddisfatte numerose relazioni temporali. La maggior parte di queste è controllata internamente dal microprocessore, cosicché, tra i diversi dispositivi del sistema, il flusso dei segnali risulta coordinato. I bus degli indirizzi, dei dati e di controllo sono così tutti legati a delle ben precise relazioni temporali.

La Figura 10-12 riporta le temporizzazioni nel caso di un'operazione di scrittura. Prima che possa essere iniziata una qualunque operazione, l'indirizzo deve essere stabile per un certo intervallo di tempo (detto *tempo di accesso*, Access Time) in modo da permettere ai decoder degli indirizzi, interni alla memoria, di selezionare le celle di memoria richieste. Il dato deve poi essere stabile per un certo intervallo di tempo (detto *tempo di set-up*, cioè di preparazione) prima che venga effettuata la scrittura vera e propria.

Il dato deve essere stabile anche per un intervallo di tempo (detto *tempo di hold*, cioè di mantenimento) dopo che è terminato l'impulso di scrittura. Da ultimo, poi, l'impulso di scrittura deve avere una durata superiore ad un certo minimo.

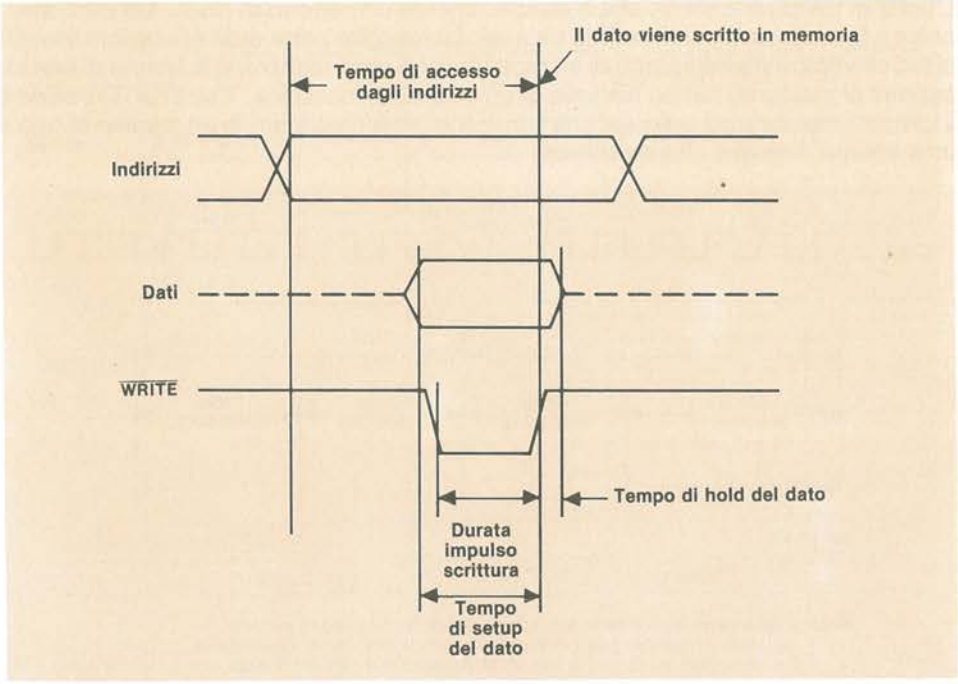


Figura 10-12. I dati sono caricati in memoria sul fronte di salita del segnale $\overline{WRITE}$

La Figura 10-13 riporta le temporizzazioni nel caso di una operazione di lettura. Come per l'operazione di scrittura, l'indirizzo deve essere stabile per un tempo abbastanza lungo perché i decoder interni alla memoria possano assestarsi. Viene quindi generato un impulso di lettura e, dopo un certo intervallo di tempo (detto tempo di *accesso del dato*, Data Access Time), la memoria presenterà sul bus dei dati il dato

indirizzato. Tale dato deve rimanere stabile per un tempo di set-up prima del fronte di salita di $\overline{READ}$, istante in cui il dato viene letto dal microprocessore. Il dato dovrà poi ancora rimanere stabile per il tempo di hold del dato.

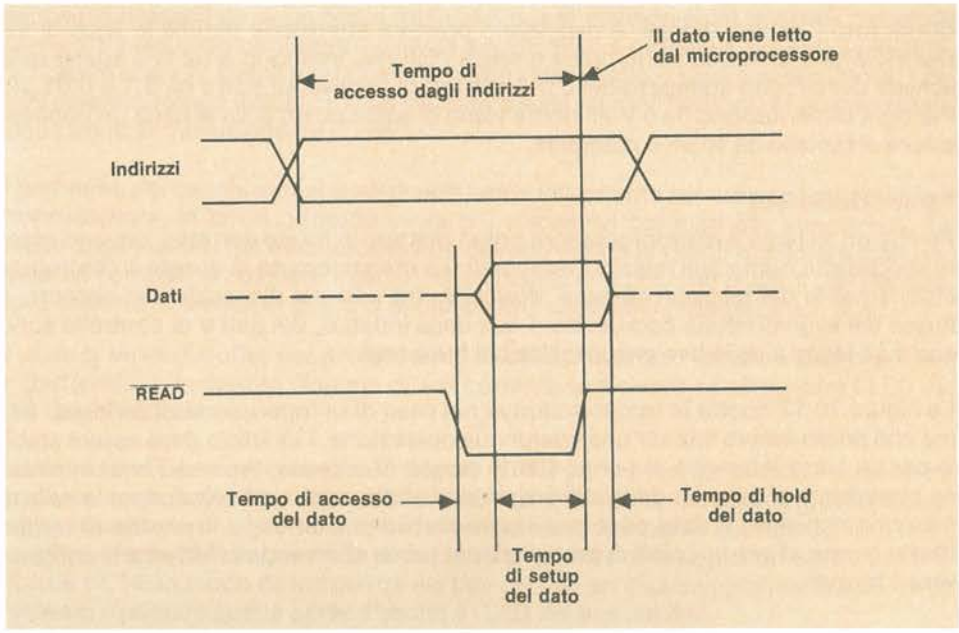


Figura 10-13. I dati vengono letti dal microprocessore sul fronte di salita del segnale $\overline{READ}$

La Figura 10-14 riporta le temporizzazioni della CPU nel caso di una tipica istruzione. L'unità di tempo è lo *stato*, che è semplicemente un periodo di clock. Un *ciclo macchina* è formato da alcuni stati, da tre a sei. La maggior parte delle operazioni semplici (ad esempio il trasferimento di un registro in un altro registro, o la lettura di una locazione di memoria) hanno bisogno di un solo ciclo macchina. Il *ciclo di istruzione* è il tempo necessario ad eseguire una istruzione, ed è costituito da un minimo di uno a un massimo di cinque cicli macchina.

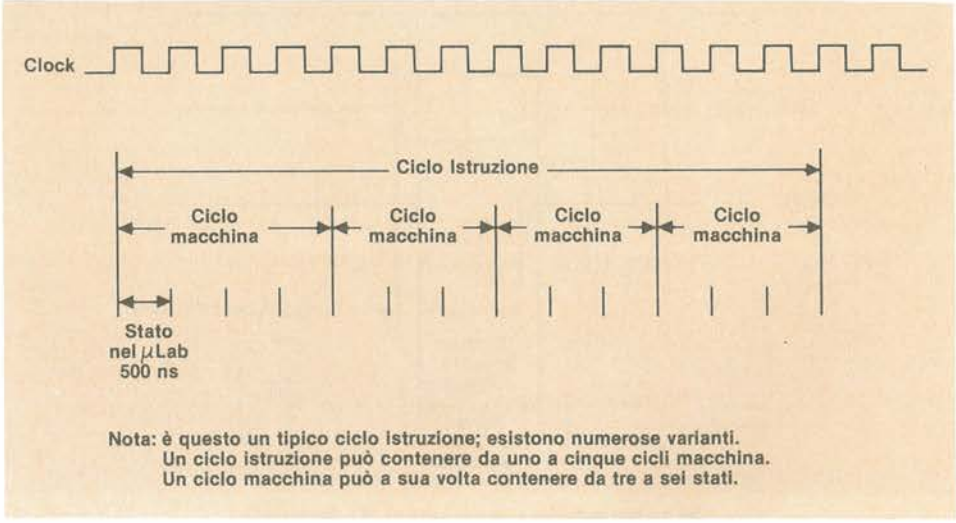


Figura 10-14. Ciclo d'istruzione tipico

La Figura 10-15 mostra il diagramma dei tempi completo nel caso di una istruzione di OUT. Gli stati sono indicati con T1, T2, ecc., mentre i cicli macchina con M1, M2, ecc. Questo diagramma sintetizza tutte le relazioni temporali già descritte: il bus in multiplex, i segnali di lettura e quelli di scrittura.

Nel primo ciclo macchina (M1) viene prelevato dalla memoria il codice operativo. Durante M2 viene letto dalla memoria il secondo byte dell'istruzione (l'indirizzo della porta). Nel corso di M3 infine l'istruzione viene eseguita: il dato viene scritto nella porta d'uscita.

Tutte le istruzioni richiedono un ciclo macchina per prelevare il codice operativo. Nel caso di istruzioni semplici, che non facciano riferimenti alla memoria o a dispositivi di I/O (ad esempio le istruzioni di trasferimento tra registri), l'esecuzione è effettuata durante il primo (ed unico) ciclo macchina. Nel caso di istruzioni a più byte (ad esempio MVI A,7 o STA 0837) è necessario un ciclo macchina per ogni byte di istruzione che deve essere prelevato. Se inoltre l'istruzione comporta un riferimento in memoria o ad un dispositivo di I/O, l'esecuzione della stessa richiederà un ulteriore ciclo. Alcune operazioni, particolarmente complesse, perché l'istruzione venga eseguita, hanno poi bisogno di un ciclo macchina supplementare, anche se l'operazione viene svolta internamente al microprocessore.

Il μLab si serve di questi cicli per i modi single-step. Il tasto HDWR STEP fa sì che venga eseguito un solo ciclo macchina per volta, mentre il tasto INSTR STEP fa sì che venga eseguito un ciclo istruzione completo.

Benché i dettagli delle temporizzazioni possano variare a seconda delle diverse situazioni, le considerazioni finora svolte sono da considerarsi fondamentali. I produttori di microprocessori hanno studiato attentamente e nei più piccoli dettagli tutte le temporizzazioni dei diversi circuiti appartenenti alla stessa famiglia, cosicché questi dispositivi sono, per quanto riguarda tale problema, tutti compatibili. È invece necessario prestare particolare attenzione quando si vuole interfacciare un certo microprocessore con dei dispositivi non specializzati.

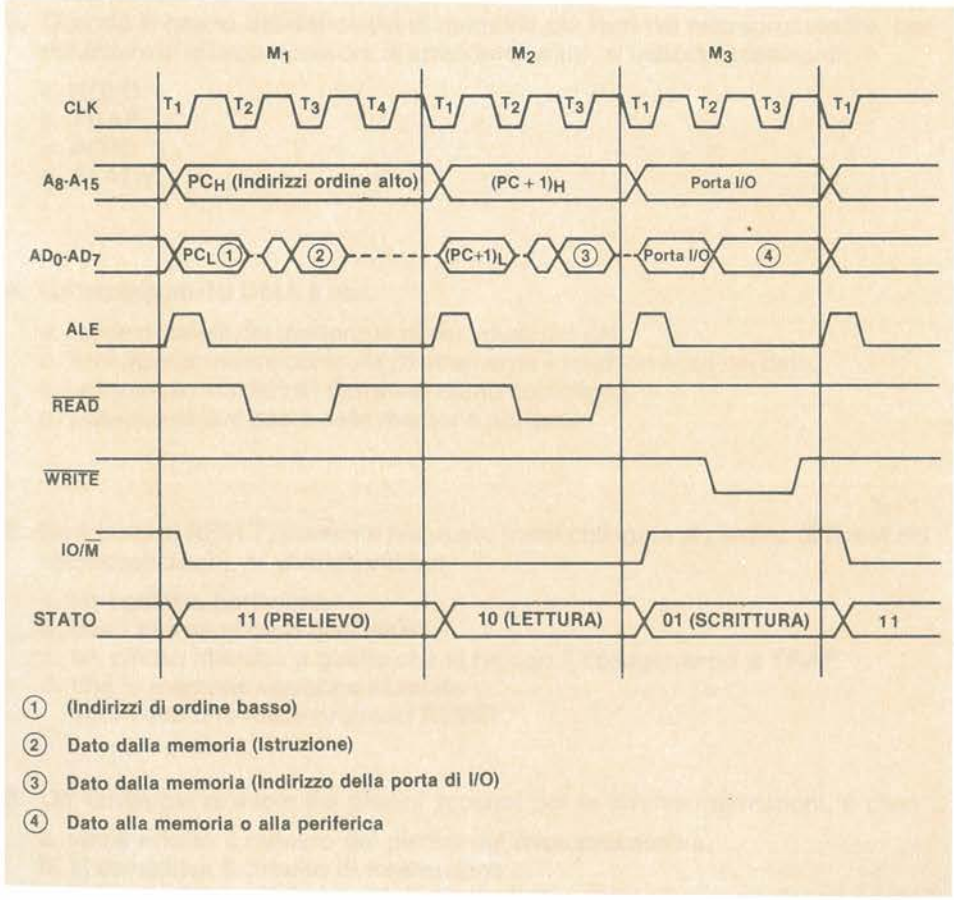


Figura 10-15. Temporizzazioni delle fasi di prelievo e di esecuzione dell'istruzione OUT

RIPASSO

Lezione 10

La porta di controllo presente nel μLab è controllata dal microprocessore ed è utilizzata per i circuiti di protezione della memoria e di single-step. Il bus in multiplex fa sì che otto piedini del microprocessore siano condivisi dal bus dei dati e dal bus degli indirizzi. La presenza, su tali linee, di una informazione di indirizzo è indicata da un'altra linea (ALE). Il piedino di Reset è utilizzato per inizializzare il sistema all'accensione. I due piedini di stato servono invece a presentare in anticipo delle informazioni di controllo. Per forzare il microprocessore ad attendere dei dispositivi lenti, si usa infine la linea Ready.

Trasferimenti di dati ad alta velocità, tra la memoria ed un circuito esterno, sono resi possibili dalla tecnica dell'accesso diretto in memoria, mediante la quale viene di fatto «scavalcato» il microprocessore. Grazie al meccanismo delle interruzioni, l'hardware esterno può richiedere che il microprocessore esegua delle particolari operazioni. Nel progetto e nel collaudo dei sistemi a microprocessore sono fattori importanti i problemi dei carichi elettrici, del bypass e delle temporizzazioni.

1. Uno dei vantaggi di avere, nel caso dell'8085, un bus in multiplex, è che:
   a. il sistema è più veloce.
   b. sono disponibili al processore più piedini per altre funzioni.
   c. si semplifica l'hardware di controllo.
   d. tutto quanto detto nei punti precedenti.

2. Quando la linea $\overline{\text{WRITE}}$ passa dal livello alto a quello basso, il bus dei dati/indirizzi contiene:
   a. un dato valido.
   b. un indirizzo valido.
   c. dei segnali di controllo validi.
   d. un'informazione non definita e instabile.

3. Quando si hanno dei dispositivi di memoria più lenti del microprocessore, per richiedere al microprocessore di attendere (wait), si utilizza il piedino di:
   a. HOLD.
   b. TRAP.
   c. INTR.
   d. READY.

4. Un vantaggio del DMA è che:
   a. rende possibili dei trasferimenti più veloci dei dati.
   b. il microprocessore controlla direttamente il trasferimento dei dati.
   c. i circuiti per trasferire i dati sono meno complicati.
   d. possono essere usate delle memorie più lente.

5. Se il tasto di RESET, presente nel μLab, fosse collegato al piedino di Reset del microprocessore, si verificherebbe:
   a. un conflitto hardware.
   b. che i bus sarebbero disabilitati.
   c. un effetto identico a quello che si ha con il collegamento a TRAP.
   d. che la memoria verrebbe azzerata ogni volta che fosse premuto RESET.

6. Un vantaggio di avere dei piedini separati per le diverse interruzioni, è che:
   a. viene ridotto il numero dei piedini del microprocessore.
   b. si semplifica il circuito di interruzione.
   c. le interruzioni non devono definire l'indirizzo della relativa routine di gestione.
   d. tutto quanto detto nei punti precedenti.

Lezione 10

7. Il motivo principale, per cui alle diverse linee di interruzione sono assegnate delle priorità, è quello di:
   a. selezionare l'indirizzo della routine d'interruzione.
   b. determinare quali sono le interruzioni più spesso usate.
   c. specificare quale interruzione debba essere selezionata nel caso se ne verifichi più di una contemporaneamente.
   d. far sì che il microprocessore non esegua più di una routine d'interruzione contemporaneamente.

8. In un sistema costituito soprattutto da dispositivi MOS, il principale carico sul bus sarà probabilmente:
   a. capacitivo.
   b. resistivo.
   c. dovuto alle correnti.
   d. dovuto alle cariche statiche.

IV
SOFTWARE DEI MICROPROCESSORI

Questa sezione presenta in modo più dettagliato il software dei microprocessori. Viene discussa una selezione delle istruzioni più rappresentative dell'8085, e vengono fatti degli esempi e introdotti degli esperimenti che ne mostrano l'uso. La Lezione 13 descrive come viene sviluppato il software e viene presentato un esempio di programma. La lezione successiva illustra i programmi per il controllo della tastiera e del display. Da ultimo vengono descritte le tecniche per rappresentare numeri molto grandi o frazionari e quelle utilizzate per calcolare funzioni matematiche complesse.

Per individuare i guasti nei sistemi a microprocessore, non è necessario avere una conoscenza dettagliata del software; ne è tuttavia opportuna una buona conoscenza generale per comprendere completamente il sistema. Perciò, anche se è possibile saltare questa sezione senza perdere la continuità del discorso, è peraltro più opportuno studiarla in modo da avere una conoscenza più completa dei sistemi a microprocessore.

LEZIONE 11

Registri e breakpoint

INTRODUZIONE

Questa lezione vi dà le nozioni di base per poter affrontare le discussioni più dettagliate, sul software dei microprocessori, che saranno presentate nelle lezioni successive. Vengono riassunte le istruzioni che sono già state descritte e ne vengono introdotte alcune altre. Sono descritti i registri del microprocessore e viene utilizzato il tasto FETCH REG del μLab che permette di esaminarli e modificarli. Da ultimo viene descritto l'uso del breakpoint, come strumento per il collaudo del software.

ISTRUZIONI GIA' DESCRITTE

Molte delle istruzioni dell'8085 sono già state descritte. In questa lezione vengono riesaminate queste istruzioni, per dare una buona base dalla quale partire per descrivere alcune nuove istruzioni. Per una descrizione completa rinviamo all'Appendice B.

Per descrivere le istruzioni, è utile adottare delle abbreviazioni. Così, nel testo seguente, il termine «dato» verrà usato per indicare una qualsiasi quantità di otto bit, mentre «adrs» servirà ad indicare un qualunque indirizzo di sedici bit.

Manipolazione dei dati: MVI, INR, CMA
Una delle operazioni fondamentali del microprocessore è quella di effettuare il caricamento dell'accumulatore con un dato. Questa operazione viene effettuata dall'istruzione *MVI A, dato* (Move Immediate to Accumulator). Il dato che deve essere posto nell'accumulatore è memorizzato nel byte che segue il codice operativo.

Una volta che il dato si trova nell'accumulatore, occorrono delle istruzioni che ne permettano la manipolazione. Le istruzioni che sono state usate fino ad ora sono *INR A* (Increment Accumulator, incrementa l'accumulatore) e *CMA* (Complement Accumulator, complementa l'accumulatore).

Controllo e salto: CPI, JMP, JZ
Per controllare il valore dell'accumulatore, può essere utilizzata l'istruzione *CPI dato* (Compare Immediate). Essa confronta il dato contenuto nel secondo byte dell'istruzione con il contenuto dell'accumulatore e altera, in funzione del risultato del confronto, il flag del processore. L'unico flag da voi finora utilizzato è stato il flag di zero, che viene posto ad «uno» se il risultato di un'operazione è zero. L'istruzione *JZ adrs* (Jump If Zero, salta se zero) fa appunto riferimento al flag di zero (che può essere stato posto a uno da una precedente istruzione, ad esempio CPI) e determina un salto se il flag è a uno (in inglese: set). Esiste anche un'istruzione di salto incondizionato *JMP adrs* che fa sì che venga effettuato un salto, non tenendo conto dello stato dei flag. Per entrambe le istruzioni di salto, l'indirizzo dello stesso è contenuto nei due byte di memoria che seguono il codice operativo.

Memoria e I/O: LDA, STA

Le istruzioni *LDA adrs* e *STA adrs* (Load Accumulator, carica nell'accumulatore e Store Accumulator, memorizza l'accumulatore) trasferiscono i dati tra l'accumulatore e la memoria o le porte di I/O. L'indirizzo della locazione di memoria, o della porta di I/O, è specificato dai due byte che seguono il codice operativo.

Subroutine: CALL, RET

Per utilizzare le subroutine occorrono due altre istruzioni. La *CALL adrs* (chiama adrs) è usata per saltare ad una subroutine, mentre la *RET* (Return, ritorno) è usata per terminare una subroutine. All'interno dell'istruzione CALL è contenuto un indirizzo, esattamente come per le istruzioni di salto. L'istruzione RET non specifica invece alcun indirizzo, ma fa sì che venga effettuato un salto all'istruzione che segue l'istruzione CALL precedentemente eseguita.

Controllo delle interruzioni: SIM, EI, DI

Per controllare le interruzioni sono disponibili tre istruzioni. *SIM* (Set Interrupt Mask, predisponi la maschera di interruzione) è usata per specificare quale interruzione dovrebbe essere abilitata e quale non lo dovrebbe. Essa copia il contenuto dell'accumulatore nel Registro Maschera delle Interruzioni, interno al processore. *EI* (Enable Interrupt, abilita le interruzioni) abilita le interruzioni specificate dal Registro Maschera delle Interruzioni. *DI* (Disable Interrupt, disabilita le interruzioni) disabilita invece tutte le interruzioni.

PERCHE' SI HANNO PARECCHI TIPI DI ISTRUZIONI

L'insieme, relativamente piccolo, delle istruzioni presentate finora illustra buona parte delle capacità di base dell'8085. Quando vi sarete familiarizzati con altre istruzioni, vi renderete conto che ci sono altre istruzioni altrettanto essenziali. Avere numerose istruzioni disponibili rende più facile scrivere dei programmi, in quanto diventa possibile scegliere tra diverse alternative. È immediata l'analogia con la progettazione hardware: anche in questo campo, esiste la possibilità di costruire un qualunque circuito logico utilizzando solo dei dispositivi NAND. In effetti sono stati realizzati in questo modo dei calcolatori completi. Tuttavia, il progetto si semplifica notevolmente se si usano anche altri dispositivi logici: NOR, flip-flop, multiplexer, contatori e sommatori.

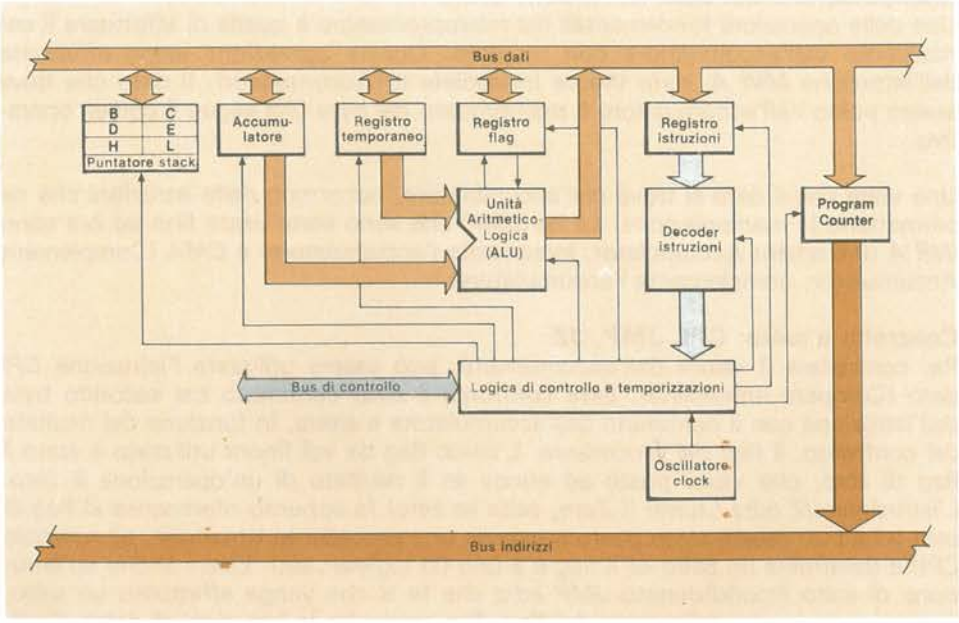


Figura 11-1. Schema a blocchi semplificato dell'8085 in cui sono evidenziati i registri di utilizzo generale

I REGISTRI DI UTILIZZO GENERALE

Fino ad ora è stata ignorata una delle principali caratteristiche del microprocessore 8085: i registri di utilizzo generale (General Purpose Registers). All'interno dell'8085 ci sono sei registri ad otto bit, che possono essere utilizzati per memorizzare temporaneamente qualunque dato. La Figura 11-1 riporta lo schema a blocchi dell'8085, in cui sono indicati tali registri, denominati registri B, C, D, E, H, e L. È pure mostrato il puntatore dello stack.

Per usare questi registri occorrono delle nuove istruzioni. L'istruzione MVI, che carica un dato nell'accumulatore, può essere in realtà utilizzata con qualunque registro. Così, *MVI D, dato* pone il dato nel registro D. La forma generale di questa istruzione è *MVI r, dato*, dove *r* indica uno qualunque dei registri (A, B, C, D, E, H o L). Sebbene l'accumulatore sia un registro di tipo un po' particolare, essendo usato per i risultati dei calcoli, può anche essere usato come registro di utilizzo generale.

Anche l'istruzione INR può essere usata con tutti i registri. La sua forma generale è *INR r*. Per esempio, INR H incrementa il registro H.

Ora che abbiamo tutti i registri, è utile avere delle istruzioni che possano muovere i dati da un registro a un altro. La forma generale di tali istruzioni è *MVI r1, r2.r1* specifica il registro destinazione, mentre *r2* specifica il registro sorgente. Per esempio, MOV A,H muove, cioè trasferisce il contenuto del registro H nell'accumulatore. Notate che la sorgente e la destinazione sono indicate nell'ordine inverso a quello che vi potreste aspettare. Vi conviene pensare alla istruzione MOV A, H come ad un'istruzione che «sposta nell'accumulatore il contenuto del registro H».

Nell'Appendice B sono riportate le istruzioni dell'8085 in forma mnemonica ed in forma esadecimale. Vi vengono così mostrate anche tutte le diverse istruzioni MOV. Queste tabelle sono la vostra guida per tradurre i codici mnemonici del linguaggio di assemblaggio in codice macchina esadecimale e viceversa. Queste tabelle sono anche un comodo elenco di tutte le istruzioni disponibili.

INDIRIZZAMENTO INDIRETTO

I registri general-purpose sono utili quando, in un programma, vengono utilizzate numerose variabili differenti. Ciascun registro può essere utilizzato per uno scopo diverso. Fino a quando, per memorizzare i dati, sono sufficienti sei registri, non sono necessarie altre locazioni RAM. Per esempio, un programma che conti sei eventi diversi, può utilizzare un registro per ciascun contatore di eventi.

Osservate che nell'elenco delle istruzioni riportate nell'Appendice B, è presente un registro (istruzioni di MOV) indicato con *M*. Non si tratta in realtà di un registro ma della locazione di memoria il cui indirizzo è contenuto nei registri H e L. I registri H e L contengono un indirizzo che punta alla locazione di memoria. Questo metodo di indirizzamento è conosciuto come *indirizzamento indiretto* (Indirect Addressing); significa cioè che l'istruzione contiene non l'indirizzo effettivo, ma l'indicazione di dove l'indirizzo stesso è contenuto (in questo caso i registri H e L).

Per esempio supponete che H contenga 12 ed L contenga 37 (Figura 11-2). L'istru-

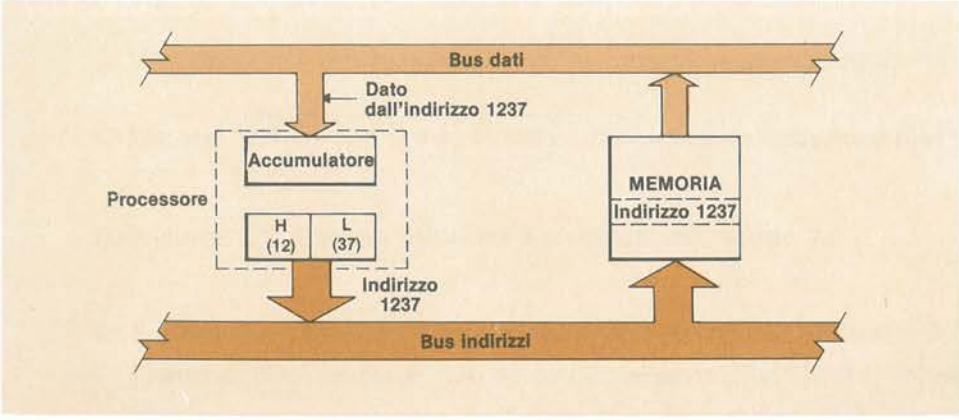


Figura 11-2. Indirizzamento indiretto attraverso i registri H e L

zione MOV A,M trasferisce così il contenuto dell'accumulatore nella locazione di memoria 1237. L'effetto è esattamente uguale a quello che si avrebbe con l'istruzione LDA 1237.

Abbiamo così un esempio di come una stessa operazione possa essere eseguita in due modi diversi. MOV A,M è un'istruzione che utilizza un solo byte, ma richiede che i registri H e L siano stati caricati con l'indirizzo desiderato. LDA 1237, d'altro canto, è un'istruzione che utilizza tre byte. Tuttavia quest'ultima è spesso preferita perché non richiede che l'indirizzo venga precedentemente caricato nei registri H e L. L'indirizzamento indiretto è particolarmente utile nel caso di operazioni che facciano riferimenti a tabelle, ad esempio quando si hanno dei «table look-up», che saranno descritti nella Lezione 15.

ESPERIMENTO 11-1

Utilizzazione dei registri

INTRODUZIONE

Viene caricato un breve programma che mostra l'uso dei registri di utilizzo generale e il modo di operare dell'istruzione MOV. Per seguire l'esecuzione del programma viene usato il tasto FETCH REG.

PROCEDIMENTO

A) Caricate da tastiera il programma della Tabella 11-1. Per prima cosa questo programma pone 37 nel registro B e poi ricopia tale valore nel registro H. Da ultimo viene incrementato il registro H stesso.

Indirizzo	Contenuto	Istruzione	Commenti
0800	06	MVI B,37	;Muovi 37 in B
0801	37		
0802	60	MOV H,B	;Muovi B in H
0803	24	INR H	;Incrementa H

Tabella 11-1. Programma dimostrativo sui registri

B) Verificate che il programma sia stato correttamente memorizzato.

C) Premete [FETCH ADRS] [0] [8] [0] [0] [INSTR STEP]. Viene eseguita l'istruzione MVI B, 37.

D) Premete [FETCH REG]. Il display visualizza il contenuto del registro A.

E) Premete due volte [STORE /INCR]. Il display visualizza il contenuto del registro B (37).
Premendo [FETCH REG] mettete il μLab nel «modo registri». I tasti [STORE /INCR] e [DECR] vengono usati poi per selezionare un registro particolare. La Tabella 11-2 presenta l'ordine in cui i registri vengono visualizzati.

ESPERIMENTO 11-1

(continuazione)

Abbreviazione	Descrizione
A	Accumulatore
FL	Flag
B, C, D, E, H, L	Registri di utilizzo generale
SPH	Byte di ordine alto del puntatore stack
SPL	Byte di ordine basso del puntatore stack
PCH	Byte di ordine alto del program counter
PCL	Byte di ordine basso del program counter
I	Stato interruzioni

Tabella 11-2. Registri dell'8085

F) Premete per cinque volte [DECR]. Il display visualizza PCH (il byte di ordine alto del program counter). Dal momento che PC è lungo sedici bit, deve essere visualizzato in due parti.

G) Per vedere il byte di ordine basso di PC, premete [STORE /INCR]. Il program counter contiene l'indirizzo della prossima istruzione che deve essere eseguita (0802).

H) Torniamo ora all'esecuzione del programma. Per riportare l'indirizzo a 0802, premete [FETCH PC]; premete poi il tasto [INSTR STEP] per eseguire l'istruzione MOV H, B.

I) Premete [FETCH REG], in modo da porre il μLab nel modo registri. Premete quindi due volte [STORE /INCR], per visualizzare il registro B. Tale registro non viene modificato dall'istruzione MOV H,B per cui contiene il valore 37.

J) Premete quattro volte [STORE /INCR], in modo da osservare il registro H. Esso contiene ora 37, il dato proveniente dal registro B.

K) Per ritornare al programma, premete [FETCH PC]. Premete poi [INSTR STEP] per eseguire l'istruzione INR H.

L) Premete [FETCH REG] ed usate ancora [STORE /INCR] per osservare il contenuto del registro H. Il valore in esso contenuto è stato incrementato dall'istruzione INR H.

RIASSUNTO

È stato introdotto un programma che carica nel registro B il valore 37, copia tale dato nel registro H ed incrementa quindi questo ultimo. Avete poi eseguito passo passo il programma ed avete verificato il modo di operare di ciascuna istruzione.

Per vedere il contenuto dei registri nell'8085, è stato usato il tasto FETCH REG in combinazione con i tasti STORE/INCR e DECR. È così possibile seguire il funzionamento di ciascuna istruzione, facendo eseguire il programma passo passo ed esaminando, dopo la esecuzione delle singole istruzioni, i registri significativi.

BREAKPOINT

Eseguendo il programma passo passo vi è possibile fermarvi tra le istruzioni ed osservare gli effetti di ciascuna di esse. Tuttavia, in molti casi è più conveniente fare girare il programma alla velocità normale di esecuzione e fermarlo invece in qualche punto di particolare interesse. Questo può essere fatto inserendo nel programma un'istruzione che obblighi il processore a saltare dal vostro programma al programma monitor. Quando verrà eseguita questa istruzione, il processore interromperà l'esecuzione del vostro programma e ritornerà al monitor. Quando poi il controllo è ritornato al monitor, avete a disposizione tutte le sue facilitazioni (ad esempio la possibilità di esaminare o alterare registri e locazioni di memoria).

Il μLab utilizza l'istruzione di Restart 1 (RST 1) per il breakpoint (punto di fermata). Tale istruzione è simile ad un'istruzione CALL, ad esclusione del fatto che non specifica un indirizzo. «RST 1» è equivalente a «CALL 0008»; l'indirizzo 0008 è univocamente fissato dal progetto dell'8085 stesso e non può quindi essere modificato. La subroutine che inizia in questa locazione ROM salva nella RAM il contenuto dei registri e trasferisce poi il controllo al programma monitor.

UTILIZZO DEI BREAKPOINT

I breakpoint sono un valido strumento per il collaudo (Debug) dei programmi. Essi vengono inseriti in punti chiave del programma, in modo da facilitare l'operazione di verifica dello stesso. I breakpoint possono essere inseriti, anche dopo che un programma è stato scritto, sostituendo un'istruzione del programma con un breakpoint. Il breakpoint ferma il programma nel punto desiderato, per cui non avete bisogno di procedere passo passo attraverso tutte le istruzioni precedenti. Potete quindi esaminare i registri e la memoria e vedere così se il programma effettua le operazioni previste. Una volta che siete sicuri che il programma funziona correttamente, potete sostituire i breakpoint con dei NOP (o con l'istruzione che era stata a sua volta sostituita dal breakpoint).

BREAKPOINT HARDWARE

Il breakpoint usato nel μLab è di tipo software; è realizzato cioè utilizzando una istruzione di breakpoint. Esistono anche breakpoint hardware, i quali sono realizzati attraverso dei circuiti logici specializzati che controllano il bus degli indirizzi. Quando viene rilevato l'indirizzo di breakpoint, il processore, utilizzando gli ingressi di controllo, viene fermato. I breakpoint software possono essere usati solamente nei programmi residenti in RAM, dal momento che l'istruzione di breakpoint deve essere memorizzata all'interno del programma. I breakpoint hardware, viceversa, possono essere indifferentemente usati con programmi residenti in ROM o in RAM.

ESPERIMENTO 11-2

Uso dei breakpoint

INTRODUZIONE

In questo esperimento inserirete un breakpoint nel programma contatore ed esaminerete l'accumulatore ad ogni giro del loop. Per avanzare nella sequenza del programma, invece di premere ripetutamente il tasto INSTR STEP (come nell'esperimento 5-1), tutto quello che dovrete fare sarà premere il tasto RUN. Quando il μLab legge ed esegue l'istruzione di breakpoint, il programma si ferma e il controllo ritorna al monitor.

PROCEDIMENTO

A) Caricate da tastiera il programma della Tabella 11-3. È questo il programma contatore dell'Esperimento 5-1, con l'aggiunta di una istruzione di breakpoint dopo l'istruzione STA.

Indirizzo	Contenuto		Istruzione	Commenti
0804	3E		MVI A,0	;Carica A con zero
0805	00			
0806	32	LOOP:	STA 3000	;Copia A sulla
0807	00			porta di uscita
0808	30			
0809	CF		RST 1	;Punto di stacco
080A	3C		INR À	;Incrementa A
080B	C3		JMP LOOP	;Ripeti
080C	06			
080D	08			

Tabella 11-3. Programma contatore con breakpoint

B) Verificate che il programma sia stato correttamente memorizzato.

C) Premete [FETCH ADRS] [0] [8] [0] [4] [RUN]. Il microprocessore esegue il programma fino al breakpoint e cede poi il controllo al monitor. Dal momento che tutti i LED della porta d'uscita sono accesi, voi sapete che le istruzioni sono state eseguite. Il display visualizza l'indirizzo della prossima istruzione da eseguire.

D) Premete [RUN]. Il programma continua finché non viene raggiunto per una seconda volta il breakpoint. I LED della porta di uscita indicano che il loop è stato eseguito un'altra volta.

E) Premete molte volte [RUN], in modo da vedere sui LED il procedere del conteggio.

ESPERIMENTO 11-2

(continuazione)

F) Premete [FETCH REG]. Il contenuto dell'accumulatore corrisponde al valore mostrato sui LED della porta di uscita.

G) Premete [FETCH PC] [RUN]. Il programma esegue ancora una volta il loop.

H) Premete [FETCH REG]. Potete ora vedere nell'accumulatore il nuovo valore.

I) Ripetete più volte i punti G e H e verificate che i LED e l'accumulatore abbiano gli stessi valori di conteggio.

RIASSUNTO

Questo esperimento spiega come servirsi di un breakpoint per osservare il modo di operare del programma contatore. Avete inserito una istruzione di breakpoint nel punto in cui volevate fermare il programma. L'istruzione di breakpoint vi permette di analizzare il funzionamento del programma, fino al punto prescelto. In questo modo avete potuto vedere meno dettagli di quanti ne avreste osservati in single-step, ma per effettuare un giro del loop avete dovuto premere un solo tasto (RUN).

L'insieme delle istruzioni base che sono state presentate permette di effettuare un trasferimento di dati tra I/O e memoria o di modificare gli stessi, in vario modo. Le istruzioni di chiamata e ritorno da subroutine facilitano l'uso della subroutine. Il cospicuo gruppo delle istruzioni MOV permette di spostare dei dati tra i diversi registri di utilizzo generale (A, B, C, D, E, H, e L) dell'8085. Questi registri sono delle comode locazioni in cui memorizzare temporaneamente dei dati.

I breakpoint fanno sì che un programma possa essere eseguito a velocità normale e fermato quindi in un punto desiderato. Il contenuto dei registri o della memoria può essere così esaminato o modificato. In questo modo è possibile verificare il corretto funzionamento del programma, senza dover eseguire singolarmente ogni istruzione.

DOMANDE

Lezione 11

1. I registri B, C, D, E, H, e L sono principalmente usati per:
   a. indicare quale interruzione dovrebbe essere abilitata.
   b. memorizzare temporaneamente i dati.
   c. controllare lo stack.
   d. specificare la prossima istruzione.

2. Il registro «M» è:
   a. un registro di uso generale.
   b. il registro istruzioni.
   c. il program counter.
   d. la locazione di memoria indirizzata da H e L.

3. Per esaminare il contenuto del registro C, dovreste premere FETCH REG e quindi _____.

4. I breakpoint sono usati per:
   a. fermare il programma nel punto desiderato.
   b. chiamare una subroutine.
   c. gestire lo stack.
   d. eseguire singolarmente ciascuna istruzione.

LEZIONE 12

Istruzioni dell'8085

INTRODUZIONE

Questa lezione descrive alcune istruzioni addizionali per la programmazione dell'8085. Non è descritto quindi il set completo d'istruzioni, ma soltanto un gruppo rappresentativo che include istruzioni aritmetiche e logiche. Il μLab è impiegato per osservare il funzionamento dell'istruzione. Viene anche discusso l'uso dello stack.

ISTRUZIONI LOGICHE

Il «gate» (porta) logico è uno degli strumenti base più comuni dell'hardware digitale. Le funzioni base proprie dei gate sono quattro: AND, NOT, OR ed OR esclusivo. Ciascuna di queste funzioni può essere realizzata anche via software.

La funzione NOT, che sarà mostrata per esempio nell'Esperimento 4.3, può essere realizzata mediante l'istruzione *Complement Accumulator* (CMA), attraverso la quale ciascun bit dell'accumulatore viene invertito.

La funzione AND è realizzata attraverso l'istruzione *And Accumulator* (ANA r). Per esempio, ANA D permette di ottenere l'AND tra il contenuto del registro D e quello dell'accumulatore, mentre il contenuto del registro (nell'esempio, D) non viene alterato. A seconda del registro su cui si opera, all'istruzione verrà associato un diverso codice operativo.

La Figura 12-1 mostra l'equivalente circuitale dell'istruzione ANA. L'operazione di AND è effettuata bit per bit su ciascun bit dell'accumulatore. Per esempio, se A = 1011 0110 e D = 0011 1100, l'istruzione ANA D è

```
      0011 1100 (registro D)
AND   1011 0110 (accumulatore)
      ---------
      0011 0100 (accumulatore)
```

La funzione OR è realizzata mediante l'istruzione *Or Accumulator* (ORA r). Quella di OR esclusivo mediante XRA r. Il modo di operare di queste istruzioni è simile a quello dell'istruzione ANA, a parte la diversa funzione logica che viene effettuata.

Dato che non sono specificati nè l'indirizzo nè i dati, tutte queste istruzioni hanno un codice costituito da un solo byte. Potete trovare l'elenco dei codici operativi nella tabella riassuntiva delle istruzioni.

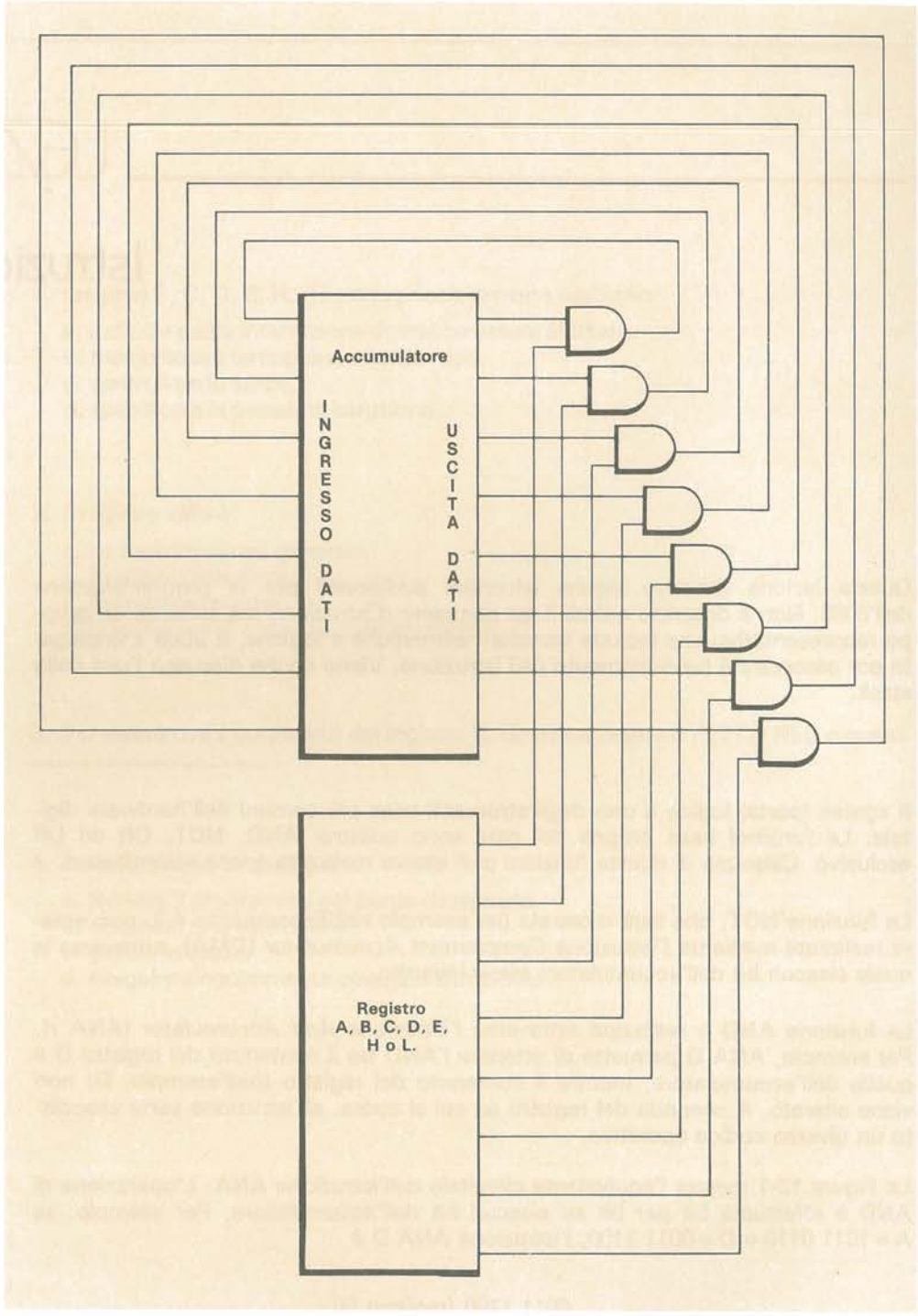


Figura 12-1. Equivalente hardware dell'istruzione AND

ESPERIMENTO 12-1

Istruzioni logiche

INTRODUZIONE

In questo esperimento dovete caricare e far girare un programma che esemplifica tre funzioni logiche: AND, OR e OR esclusivo. I dati d'ingresso sono definiti attraverso gli interruttori della porta d'ingresso. Su di essi opera l'istruzione logica. I LED connessi sulla porta di uscita visualizzano i risultati.

PROCEDIMENTO

A) Traducete il programma della Tabella 12-1 in codice macchina e caricatelo, attraverso la tastiera, nel μLab. Ogni codice mnemonico deve essere convertito nel codice operativo corretto (ricavato dalla tabella in appendice delle istruzioni) e seguito dai dati richiesti (se ci sono). Il listing completo del programma è riportato in Appendice A.

Questo programma legge un dato dagli interruttori della porta d'ingresso e realizza l'AND tra tale dato e il valore 0011 1100. Il risultato è scritto sui LED della porta d'uscita. Il programma ritorna poi all'inizio, in modo che voi possiate cambiare l'ingresso e verificare la risposta d'uscita.

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	_____	START:	LDA 2000	;Leggi la porta d'ingresso
0801	_____			
0802	_____			
0803	_____		MVI B,3C	;Carica il registro B con 0011 1100
0804	_____			
0805	_____		ANA B	;AND dell'accumulatore con B
0806	_____		STA 3000	;Invia il risultato alla porta di uscita
0807	_____			
0808	_____			
0809	_____		JMP START	;Ripeti
080A	_____			
080B	_____			

Tabella 12-1. Programma per spiegare le istruzioni logiche

B) Verificate che il programma sia correttamente memorizzato.

C) Premete [FETCH ADRS] [0] [8] [0] [0] [RUN]. Il programma sta ora girando.

D) Ponete tutti gli interruttori d'ingresso a «1» (alto). I LED di uscita visualizzano la configurazione 0011 1100 (0 = acceso = on, 1 = spento = off).

(continuazione)

E) Modificate le posizioni degli interruttori di ingresso e osservate come si comportano i LED d'uscita. I bit che vengono posti in AND con uno «zero» non saranno influenzati dagli interruttori di ingresso.

F) Premete [RESET] per ridare il controllo al monitor. Notate che gli interruttori d'ingresso non influenzano più i LED di uscita. Cambiate l'istruzione ANA B (all'indirizzo 0805) con ORA B (codice operativo B0).

G) Fate partire il programma.

H) Osservate attentamente il comportamento dei LED di uscita, in funzione del valore impostato sugli interruttori. Osservate la differenza tra le funzioni AND ed OR. I bit che sono in OR con gli zero sono interessati dai commutatori d'ingresso.

I) Premete [RESET]. Cambiate l'istruzione ORA B con XRA B (codice operativo A8).

J) Ripetete i punti G e H. I bit che sono in OR esclusivo con gli uno sono invertiti.

RIASSUNTO

In questo esperimento il μLab legge diversi valori dagli interruttori di ingresso e visualizza il risultato di diverse istruzioni logiche sui LED di uscita: potete così osservare come operano le istruzioni logiche. Utilizzando la funzione AND potete mettere a zero alcuni bit selezionati. Con la funzione OR potete invece mettere a 1 alcuni bit. Con XOR infine, alcuni bit possono essere complementati.

MASCHERATURA

Utilizzando istruzioni logiche, è possibile effettuare operazioni di *mascheratura*, selezionare cioè alcuni bit all'interno di una parola. Supponiamo, per esempio, di avere otto interruttori connessi ad una porta d'ingresso. Per poter analizzare un solo bit (un solo interruttore) il processore deve trascurare tutti i rimanenti bit. La Figura 12-2 mostra il diagramma di flusso di un programma che effettua un test sull'interruttore collegato al bit 3 della porta di ingresso. Se l'interruttore è aperto (off), i LED di uscita vengono tutti accesi, altrimenti vengono spenti.

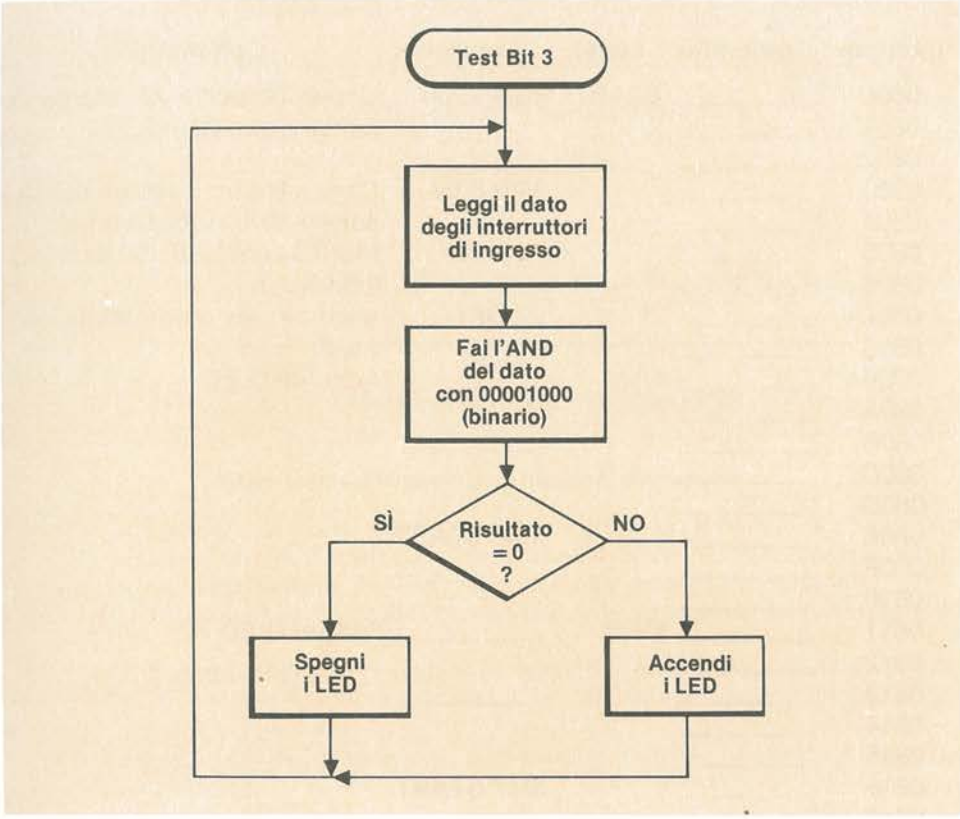


Figura 12-2. Diagramma di flusso del programma che effettua il test del bit 3 della porta di ingresso

La Tabella 12-2 mostra il programma. Esso legge per prima cosa i dati dalla porta d'ingresso e li pone nell'accumulatore. Poi predispone nel registro B il valore di *maschera* e ne esegue l'AND con l'accumulatore. Si ottiene che tutti i bit, eccetto il bit 3, sono forzati a zero. (Siccome 0 AND 1 = 0, e 1 AND 1 = 1, qualsiasi bit posto in AND con 1 rimane immutato). Per ultimo il programma utilizza l'istruzione JZ per eseguire un salto se il flag di zero è posto a uno, se cioè l'intero byte (perciò anche il bit 3) è zero.

ESERCIZIO DI PROGRAMMAZIONE N. 1: MASCHERATURA

Completate il programma della Tabella 12-2, aggiungendo le routine di ON e OFF. La routine di ON verrà utilizzata per accendere i LED della porta di uscita e quella di OFF per spegnerli.

Traducete il programma in codice macchina e caricatelo da tastiera nel μLab in un'area qualsiasi tra 0800 e 0B00.

Analizzate attentamente il vostro programma, e verificate che sia memorizzato correttamente. Fate girare il programma e verificate che gli interruttori di ingresso controllino in modo corretto i LED di uscita: i LED si devono accendere solamente quando l'interruttore corrispondente al bit 3 è alto. La posizione degli altri interruttori non dovrebbe invece avere alcuna importanza.

Ora modificate il programma in modo da poter effettuare il test su un diverso bit della porta (un diverso interruttore). Se non dovesse funzionare correttamente, ricontrollate la logica del programma e la traduzione in codice macchina. Poi verificate se il codice è correttamente memorizzato. Infine se proprio non avete altrimenti risolto il problema andate a vedere l'Appendice A dove è riportato l'intero programma.

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	_____	START:	LDA 2000	;Copia la porta di ingresso nell'accumulatore
0801	_____			
0802	_____			
0803	_____		MVI B,08	;Carica B con il valore di maschera (00001000 binario)
0804	_____			
0805	_____		ANA B	;Metti a zero tutti i bit escluso il bit 3
0806	_____			
0807	_____		JZ OFF	;Verifica se l'accumulatore è = 0
0808	_____			
0809	_____	ON:	_____	;Accendi i LED
080A	_____			
080B	_____		_____	
080C	_____			
080D	_____			
080E	_____		JMP START	
080F	_____			
0810	_____			
0811	_____	OFF:	_____	;Spegni i LED
0812	_____			
0813	_____		_____	
0814	_____			
0815	_____			
0816	_____		JMP START	
0817	_____			
0818	_____			

Tabella 12-2. Programma test bit 3

AZZERAMENTO DELL'ACCUMULATORE

L'istruzione XRA A esegue l'OR esclusivo tra l'accumulatore e se stesso. Dal momento che l'OR esclusivo di qualunque dato con se stesso è zero, questa operazione azzera l'accumulatore. Questo è in effetti il modo migliore per azzerare l'accumulatore dal momento che l'istruzione alternativa, MVI A, 0 richiede due byte di memoria, a differenza del solo byte richiesto da XRA A (codice operativo AF).

OPERAZIONI DI SHIFT

Un'altra funzione comunemente richiesta è quella di shift (scorrimento) di un dato verso destra o verso sinistra. Questa operazione è realizzata, circuitalmente, utilizzando un registro di shift. Lo 8085 è dotato di istruzioni che permettono lo shift del dato contenuto nell'accumulatore; ad esempio *Rotate Right Circular* (RRC, ruota a destra in modo circolare) realizza uno shift verso destra, mentre *Rotate Left Circular* (RLC, ruota a sinistra in modo circolare) realizza uno shift verso sinistra. Queste istruzioni operano solamente sull'accumulatore e «Circular» si riferisce al fatto che il bit meno significativo, LSB, è fatto scorrere nel bit più significativo, MSB (o viceversa) come mostrato in Figura 12-3.

Per esempio, supponiamo che l'accumulatore contenga il valore 0010 0001. Dopo l'esecuzione di RRC, conterrà 1001 0000. Notare che il bit meno significativo si è

portato nel bit più significativo. Questo si verifica perché i dati ruotano come se i bit fossero disposti in un anello, con MSB e LSB in posizioni adiacenti.

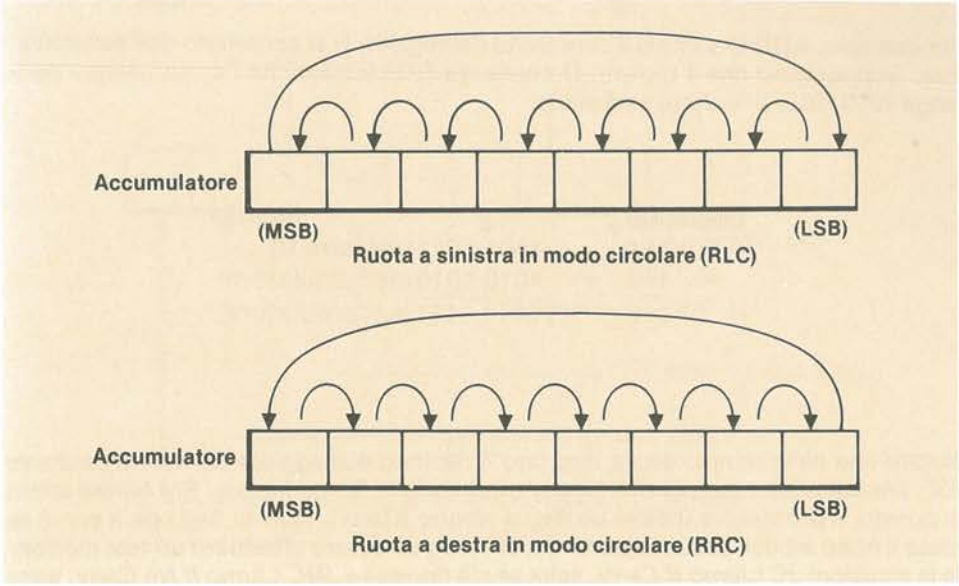


Figura 12-3. Come operano le istruzioni di rotazione

ESERCIZIO DI PROGRAMMAZIONE N. 2: ROTAZIONI

Secondo il diagramma di flusso della Figura 12-4, scrivete un programma che mostri come operano le istruzioni di rotazione. Traducete il programma in codice macchina e caricatelo, tramite la tastiera, nel μLab. Eseguite passo passo il programma, verificando che i LED della porta d'uscita si comportino coerentemente con l'istruzione di rotazione.
Aggiungete al programma un breakpoint in modo da usare il tasto RUN invece di INSTR STEP. Ogni volta che verrà premuto RUN i dati dovrebbero subire uno scorrimento.
Se non riuscite a far girare il programma, potete consultare l'Appendice A in cui è riportata la soluzione.

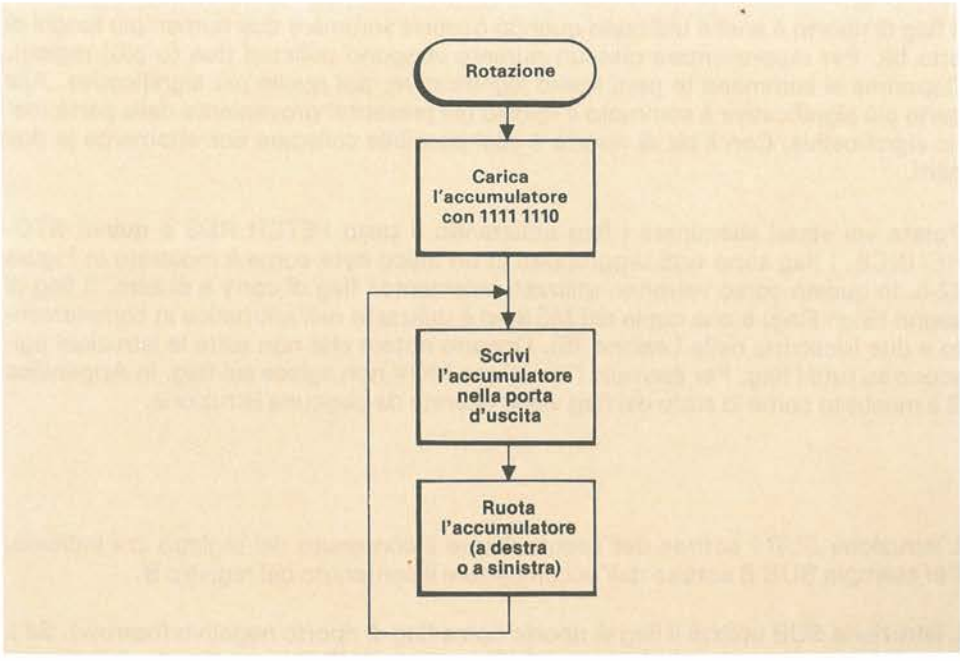


Figura 12-4. Diagramma di flusso per l'Esercizio di Programmazione N. 2

ADDIZIONE

La somma di due numeri è l'operazione aritmetica fondamentale. L'istruzione ADD r somma il contenuto del «r» registro indicato a quello dell'accumulatore e memorizza il risultato in quest'ultimo.

Per esempio, ADD D somma il contenuto del registro D al contenuto dell'accumulatore. Supponiamo che il registro D contenga 1001 0011 e che l'accumulatore contenga 1010 1010. Il risultato sarà allora:

Decimale		Binario	
147		1001 0011	(registro D)
+ 170	+	1010 1010	(accumulatore)
317		1 0011 1101	(accumulatore)

FLAG DI RIPORTO

Notate che nell'esempio sopra riportato il risultato è maggiore del valore decimale 255, per cui occorrono più di 8 bit per esprimerlo in forma binaria. Per tenere conto di questo, il processore utilizza un *flag di riporto* (Carry). Questo flag opera come se fosse il nono bit dell'accumulatore. Su di esso può essere effettuato un test mediante le istruzioni JC (*Jump If Carry*, salta se c'è riporto) e JNC (*Jump If No Carry*, salta se non c'è riporto). Questo test è analogo a quello che può essere effettuato per il flag di zero.

Il seguente programma, per esempio, somma il registro D all'accumulatore e salta alla locazione 0820 se viene generato riporto:

```
AND  D
JC   0820
```

Se non viene generato riporto (risultato < 256 decimale) il programma continuerà con l'istruzione successiva.

Il flag di riporto è anche utilizzato quando occorre sommare due numeri più lunghi di otto bit. Per rappresentare ciascun numero vengono utilizzati due (o più) registri. Dapprima si sommano le parti meno significative, poi quelle più significative. Alla parte più significativa è sommato il riporto (se presente) proveniente dalla parte meno significativa. Con il bit di riporto è così possibile collegare correttamente le due parti.

Potete voi stessi esaminare i flag utilizzando il tasto FETCH REG e quindi STORE/INCR. I flag sono tutti raggruppati in un unico byte come è mostrato in Figura 12-5. In questo corso verranno utilizzati solamente i flag di carry e di zero. Il flag di segno (Sign Flag) è una copia del MSB ed è utilizzato nell'aritmetica in complemento a due (descritta nella Lezione 15). Occorre notare che non tutte le istruzioni agiscono su tutti i flag. Per esempio l'istruzione MOV non agisce sui flag. In Appendice B è mostrato come lo stato dei flag viene alterato da ciascuna istruzione.

SOTTRAZIONE

L'istruzione *SUB r* sottrae dall'accumulatore il contenuto del registro «r» indicato. Per esempio SUB B sottrae dall'accumulatore il contenuto del registro B.

L'istruzione SUB utilizza il flag di riporto come flag di riporto negativo (borrow). Se il flag di riporto è uguale a 1 al termine dell'istruzione SUB B, vuol dire che il contenuto del registro B è maggiore di quello dell'accumulatore.

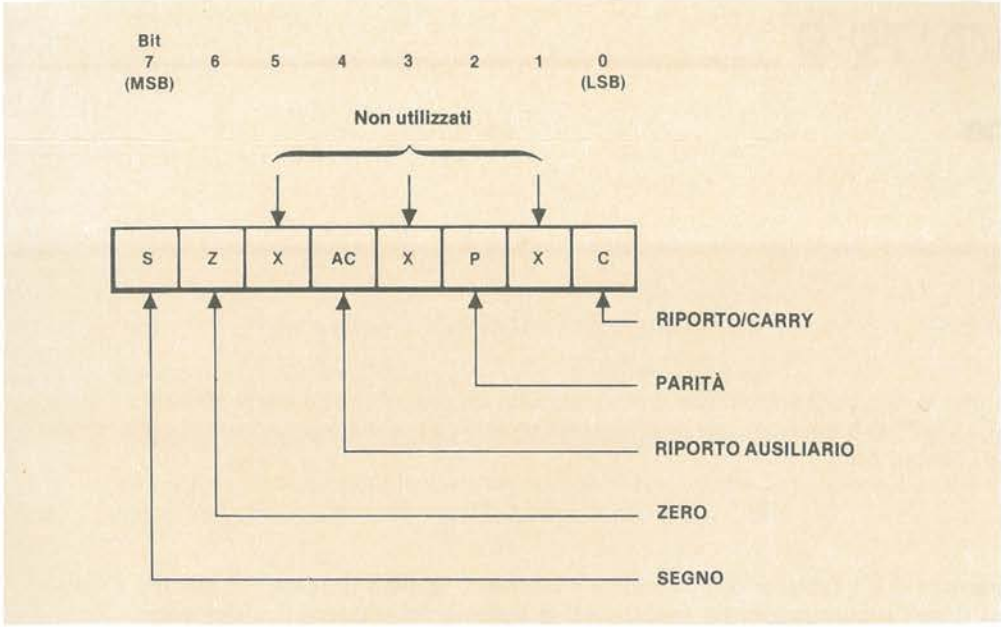


Fig. 12-5. I flag del microprocessore

ESPERIMENTO 12-2

Funzioni aritmetiche

INTRODUZIONE

Per mostrare come operano le istruzioni aritmetiche, viene eseguito un programma e viene utilizzato il tasto INSTR STEP. Il tasto FETCH REG è utilizzato per predisporre i registri prima dell'esecuzione di un'istruzione e per esaminarli dopo l'esecuzione.

PROCEDIMENTO

A) Traducete il programma della Tabella 12-3 in codice macchina. Questo programma sottrae il contenuto del registro B dell'accumulatore ed esegue poi la somma del registro C e dell'accumulatore.

Indirizzo	Contenuto	Istruzione	Commenti
0800	_____	SUB B	;Sottrai B dall'accumulatore
0801	_____	ADD C	;Somma C all'accumulatore

Tabella 12-3. Esempio di programma aritmetico

B) Caricate il programma nel μLab.

C) Premete [FETCH REG], in modo da visualizzare il contenuto dell'accumulatore.

D) Introducete il dato A7 e premete [STORE /INCR]. Questo permette di memorizzare A7 nell'accumulatore. (In realtà, il dato è memorizzato in una posizione della RAM che il monitor utilizza per salvare il contenuto dell'accumulatore. L'accumulatore verrà poi caricato dalla memoria quando verrà premuto RUN, INSTR STEP o HDWR STEP).

E) Premete [DECR] per verificare che l'accumulatore contenga A7.

F) Premete, due volte, [STORE /INCR] per visualizzare il contenuto del registro B.

G) Introducete da tastiera il valore 23. Premete [STORE /INCR] per memorizzare questo dato nel registro B.

H) Viene ora visualizzato il registro C. Introducete il dato 93. Premete per memorizzarlo nel registro C.
I registri contengono ora questi valori:
A = A7 B = 23 C = 93

I) Sono stati così predisposti tutti gli ingressi al programma. Premete [FETCH ADRS] [0] [8] [0] [0] [INSTR STEP] in modo da eseguire l'istruzione SUB B.

J) Premete [FETCH REG] per vedere il contenuto dell'accumulatore. Contiene il valore previsto? Ricordate che questi valori sono esadecimali (A7 hex — 23 hex = 84 hex).

K) Premete [STORE /INCR] per visualizzare il registro dei flag. Convertite in binario il dato visualizzato e riferendovi alla Figura 12-5 controllate il valore dei flag di riporto e di zero. Questa sottrazione non genera riporto negativo (il flag di riporto è uguale a «0»), e il risultato è zero.

L) Utilizzando [STORE /INCR] per accedere ai vari registri, verificate che B e C non sono cambiati.

M) Premete [FETCH P C] [INSTR STEP] per eseguire l'istruzione ADD C.

N) Premete [FETCH REG]. Il risultato dell'addizione è 84 hex + 93 hex = 117 hex. Dal momento che il risultato è maggiore di FF hex, l'accumulatore di 8 bit non è in grado di contenerlo completamente. Gli 8 bit meno significativi (17 hex) rimangono nell'accumulatore, ma viene generato un carry.

O) Premete [STORE /INCR] per visualizzare il registro dei flag. Controllate lo stato di riporto. Il flag è uguale a «1»; il risultato è infatti maggiore di FF hex (255 decimale) e di conseguenza è stato generato un carry.

P) Verificate che non sono cambiati i registri B e C.

RIASSUNTO

In questo esperimento sono state mostrate le istruzioni di somma e sottrazione. Al termine di ciascuna operazione, il risultato è contenuto nell'accumulatore, mentre gli altri registri contengono i valori originali. Il flag di riporto segnala se si è verificata una condizione di riporto (o di riporto negativo).

Con il tasto FETCH REG avete introdotto i dati di ingresso del programma. Avete poi esaminato le uscite del programma e così facendo avete potuto vedere direttamente il modo di operare di queste istruzioni.

SUBROUTINE E STACK

Le subroutine (sottoprogrammi) permettono di semplificare i programmi complessi, in quanto permettono loro di utilizzare una sola routine in vari punti del programma. Per esempio supponiamo che un programma debba eseguire una serie di calcoli che richiedano molte operazioni. In questo caso si può scrivere una subroutine generale di moltiplicazione, la quale viene richiamata ogni qualvolta si debba eseguire una moltiplicazione. Le subroutine vengono anche usate per suddividere il programma in tanti piccoli moduli, come è descritto nella Lezione 13. Per accedere ad una subroutine viene utilizzata l'istruzione di CALL. Quando viene eseguita l'istruzione CALL, il microprocessore salva il contenuto del program counter (indirizzo di ritorno) nello *stack*. Lo stack (pila o catasta) è una parte di memoria ad accesso speciale. Il programma si riporta a tale indirizzo di ritorno, memorizzato nello stack, quando viene eseguita, al termine della subroutine, l'istruzione RET.

Supponiamo che venga utilizzato un unico registro a 16 bit per memorizzare l'indirizzo di ritorno. Potete notare quali sono i problemi causati? Se una subroutine ne chiama un'altra (le subroutine sono allora dette nested, *nidificate*) il primo indirizzo di ritorno viene perduto.

La soluzione a questo problema è rappresentata dall'uso di un gruppo di locazioni di memoria (chiamato stack). Poiché una routine ne chiama un'altra, gli indirizzi di ritorno sono memorizzati in locazioni di memoria sequenziali. La figura 12-6 mostra la sequenza delle operazioni quando la routine A richiama la routine B che richiama la routine C. I contenuti dello stack e del PC a ciascun passo sono mostrati nella Figura 12-7. La sequenza di eventi è la seguente:

a. Il programma principale è in esecuzione.

b. Il programma principale richiama la subroutine A. L'indirizzo di ritorno è memorizzato nello stack e il PC è caricato con l'indirizzo di partenza della routine A.

c. La routine A chiama la routine B. L'indirizzo di ritorno per la routine A è collocato nello stack e il PC è caricato con l'indirizzo di partenza della routine B.

d. La routine B ritorna. L'ultimo indirizzo collocato nello stack viene caricato nel PC e il controllo ritorna alla routine A.

e. La routine A restituisce il controllo al programma principale.

Si può paragonare lo stack ad una pila di piatti in cui i piatti possono essere aggiunti o da cui in modo inverso possono essere tolti. Questo stack viene denominato *push-down* (spingi verso il basso) o *Last In First Out* (LIFO, ultimo a entrare primo ad uscire). Il microprocessore 8085 usa la RAM come stack, e quest'ultimo può quindi avere dimensioni arbitrarie. L'indirizzo della cima (Top) dello stack è contenuto in un apposito registro (Stack Pointer, puntatore dello stack) del processore. La posizione dello stack è definita introducendo nel puntatore dello stack l'indirizzo desiderato.

All'accensione del μLab il puntatore dello stack contiene 0BB0. Questo permette di utilizzare già lo stack senza doverne definire la posizione. È possibile che i vostri programmi portino il puntatore dello stack al di fuori dell'area RAM disponibile. Questo può succedere, per esempio, se sono chiamate una serie di subroutine senza che vengano eseguite le istruzioni di ritorno. Quando si verifica questa condizione il microprocessore si ferma (halt). Tuttavia la prima volta che viene premuto il tasto di RUN o STEP il monitor controlla il puntatore dello stack: se esso ha un valore non accettabile (< 0B40 o > 0BFF) viene ripristinato al suo valore iniziale. Normalmente il puntatore dello stack non supera i valori consentiti, per cui il monitor non è costretto a modificarlo.

Si può esaminare il puntatore dello stack utilizzando il tasto FETCH REG. Dal momento che si tratta di un registro a 16 bit, viene visualizzato in due parti: il byte di ordine più elevato (SPH) e quello di ordine più basso (SPL).

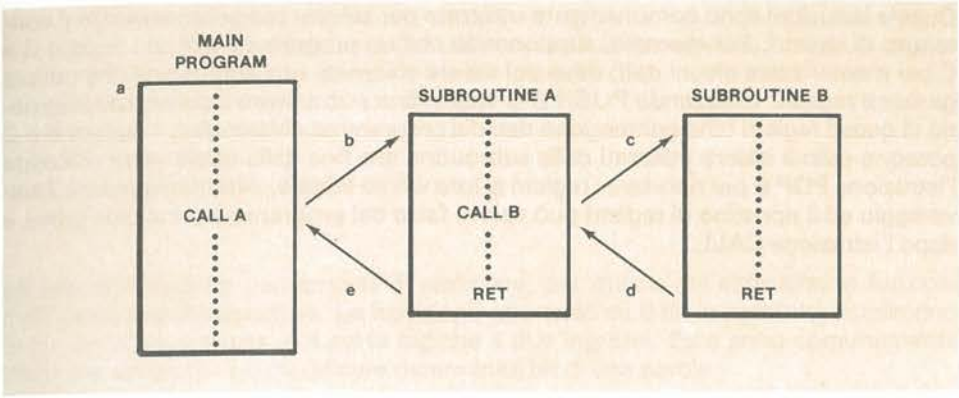


Figura 12-6. Sequenza di eventi relativi al programma principale che chiama la routine A che chiama a sua volta la routine B

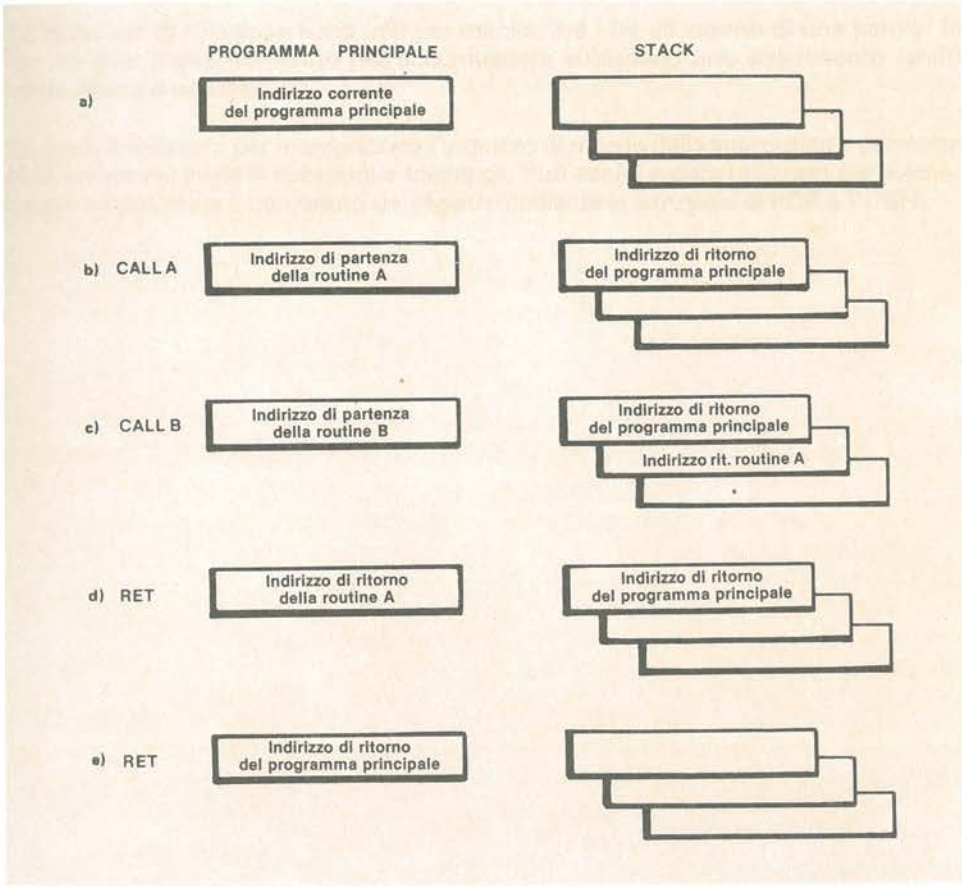


Figura 12-7. Come opera lo stack

ISTRUZIONI PUSH E POP

Le istruzioni CALL e RET usano in modo automatico il puntatore dello stack, ma è anche possibile usare lo stack in modo «manuale». L'istruzione PUSH pone il contenuto di un registro nello stack mentre l'istruzione POP riporta i dati, dallo stack, nei registri. Dal momento che lo stack è strutturato per memorizzare degli indirizzi, lunghi 16 bit, i registri saranno memorizzati a due alla volta. Per esempio, PUSH B pone i registri B e C nella cima dello stack, mentre POP B ripristina il contenuto di entrambi i registri leggendoli dalla cima dello stack.

Queste istruzioni sono comunemente utilizzate per salvare temporaneamente il contenuto di registri. Per esempio, supponendo che un programma utilizzi i registri B e C per memorizzare alcuni dati; deve poi essere chiamata una subroutine che utilizza gli stessi registri. Utilizzando PUSH B la subroutine può salvare il contenuto originario di questi registri (che contengono dati del programma chiamante). I registri B e C possono quindi essere utilizzati dalla subroutine alla fine della quale verrà utilizzata l'istruzione POP B per riportare i registri al loro valore iniziale. Alternativamente il salvataggio ed il ripristino di registri può essere fatto dal programma principale prima e dopo l'istruzione CALL.

Lezione 12

Le istruzioni logiche permettono di realizzare, per mezzo del software, le funzioni delle porte logiche standard. Le istruzioni, operando su 8 bit in parallelo, forniscono la funzione equivalente di 8 porte logiche a due ingressi. Esse sono comunemente usate per selezionare o modificare determinati bit di una parola.

Tra le istruzioni aritmetiche si hanno le istruzioni base di somma e sottrazione. Sia per le istruzioni aritmetiche che per quelle logiche, il risultato dell'operazione è posto nell'accumulatore. Il flag di riporto è usato dalle istruzioni artimetiche per indicare un overflow (traboccamento) o un borrow (riporto negativo).

Le istruzioni di rotazione sono utili per manipolare i bit all'interno di una parola. In questo caso i dati contenuti nell'accumulatore subiscono uno scorrimento (shift) verso destra o sinistra.

Lo stack è utilizzato per memorizzare l'indirizzo di ritorno delle subroutine e permette di chiamare più livelli di subroutine (nesting). Può anche essere utilizzato per memorizzare e ripristinare il contenuto dei registri mediante le istruzioni di POP e PUSH.

DOMANDE

Lezione 12

1. Supponete che il registro B contenga 37 hex e l'accumulatore contenga 14 hex. Dopo l'esecuzione dell'istruzione ANA B, l'accumulatore conterrà:

   a. 37 (hex).
   b. 14.
   c. 4B.
   d. 23.

2. Nella domanda 1, e se l'istruzione fosse ORA B, il risultato sarebbe:

   a. 37.
   b. 14.
   c. 4B.
   d. 23.

3. Per lasciare invariato il bit più significativo (MSB) e portare tutti gli altri bit a zero, l'accumulatore dovrebbe eseguire un AND con:

   a. 01.
   b. 10.
   c. 80.
   d. F1.

4. Se l'accumulatore contiene inizialmente F3 hex; il programma seguente salterà all'indirizzo_____

```
MVI   B,23.
ADD   B.
JC    0930.
JMP   0900.
```

5. Lo stack è usato per memorizzare gli indirizzi di ritorno delle subroutine, perché:

   a. la memoria normale è troppo piccola.
   b. i registri non possono contenere un indirizzo.
   c. permette di chiamare più livelli di subroutine.
   d. è vero tutto quanto detto ai punti a, b, c.

LEZIONE 13

Tecniche di progettazione software

INTRODUZIONE

Progettare un programma complesso è difficile quanto progettare un circuito complesso, ma con problemi a volte diversi. Per progettare del software è necessario un nuovo insieme di tecniche che rendano il programma facile da verificare e modificare. Questa lezione descrive come viene sviluppato del software. Come esempio, viene riportato un programma di controllo per semafori.

Questa lezione tratta solo del progetto di sistemi software. Se non siete interessati all'argomento, potete saltare questa lezione che non è necessaria alla comprensione delle lezioni successive.

SISTEMI DI SVILUPPO

Quando si scrivono dei programmi utilizzando il μLab, il linguaggio simbolico deve essere tradotto «manualmente» nel linguaggio macchina. Il codice macchina viene poi caricato nel μLab da tastiera. Se questo può essere un sistema adeguato per scrivere programmi brevi, è però troppo laborioso per la maggior parte dei progetti software.

Sistema di sviluppo Intel Intellec MDS® . *In senso orario da sinistra vediamo i seguenti elementi: terminale CRT, stampante, doppia unità floppy disk, lettore di nastro perforato, programmatore di PROM e microcalcolatore con scheda ICE. (Intel Corp.)*

La maggior parte del software per microprocessori è scritto utilizzando un *sistema di sviluppo*. Un sistema di sviluppo è un microcomputer opportunamente progettato per essere di aiuto nello sviluppo di progetti software e hardware. Un sistema tipico comprende 32 + 64 K byte di RAM, un floppy disk doppio per memorizzare programmi, un terminale video con tastiera ed una stampante. Il software comprende generalmente un programma editor per introdurre ed editare i programmi simbolici, un programma assemblatore (assembler) per tradurre il simbolico in linguaggio macchina e un monitor o «debugger» per predisporre dei punti di interruzione, esaminare i registri, ecc. Un tale sistema facilita enormemente il processo di sviluppo dei programmi.

Per aiutare la fase di collaudo dell'hardware, molti sistemi dispongono di un ICE (*In-Circuit Emulator*, emulatore inserito nel circuito). Il chip del microprocessore viene rimosso dal sistema in prova e al suo posto viene inserito un cavo proveniente dal sistema di sviluppo (con un connettore a quaranta piedini posto alla sua estremità. Il sistema di sviluppo può ora controllare completamente le operazioni del sistema in prova. L'ICE è uno degli strumenti più sofisticati di progetto e di collaudo attualmente disponibili.

COME SI PROCEDE IN UN PROGETTO SOFTWARE

Quando si progetta un programma complesso, è importante seguire una metodologia ordinata. Indichiamo di seguito alcuni passi che devono essere tipicamente effettuati:

- Definire il problema.
- Progettare la soluzione-suddivisione in blocchi funzionali.
- Scrivere il diagramma di flusso del programma.
- Codificare il programma.
- Provare e correggere ciascuna routine.
- Provare e correggere l'intero programma.

Il primo passo, pur essendo il più ovvio, è frequentemente sottovalutato. Prima di iniziare a progettare, deve essere ben definito e compreso il problema. Cioè è necessario avere un insieme di specifiche precise per il programma. Quali sono gli ingressi, come devono essere trattati, quali sono le uscite?

Una volta che tutto questo è stato stabilito, e solo allora, potete partire a implementare il software. Pensate però a tutto il progetto nel suo insieme prima di iniziare a codificare i programmi o anche a disegnare il flusso logico. È buona cosa suddividere l'intero programma in sezioni modulari relativamente piccole. Controllate che per ciascuna sezione siano stati definiti gli ingressi e le uscite correttamente. Tutto questo vi permette di affrontare il problema in blocchi più maneggevoli e al contempo vi permette di collaudare il programma a piccole parti per volta: questo vi rende molto più facile trovare errori. Lo stesso approccio viene seguito nella progettazione dell'hardware, dove il sistema viene diviso in blocchi logici funzionali.

Ci sono molti modi per organizzare un programma. Uno degli approcci migliori è conosciuto come «programmazione strutturata» (Structured Programming). Pur non essendone data, in questa sede, una descrizione dettagliata, ne verrà seguita la filosofia generale.

Una volta che il problema sia stato definito, che sia stato scelto un approccio e che il programma sia stato diviso in moduli, potete costruire il vostro diagramma di flusso (Flow-Chart). Viene in genere fatto un diagramma di flusso globale che fornisce una «fotografia panoramica» del problema e mostra ciascun modulo del programma come una singola scatola. Esisterà poi il diagramma di flusso di ciascun modulo.

Solo a questo punto si può scrivere il programma vero e proprio. Se i precedenti passi sono stati seguiti attentamente e scrupolosamente, questo lavoro diventa notevolmente facile. La stessa situazione si verifica nella progettazione hardware: una volta che è stato definito il corretto approccio al problema, l'implementazione reale è in generale relativamente facile.

I moduli di programma, dopo che sono stati scritti, devono essere provati. Anche un esperto programmatore non si aspetta che i suoi programmi funzionino subito correttamente. A questo punto diventa molto importante la modularità del programma. Ciascun modulo deve poter essere provato separatamente o insieme a routine che sono già state verificate. Vengono inseriti nel modulo i dati di ingresso e deve essere verificata la correttezza dei risultati al termine della sua esecuzione. È molto più facile correggere un singolo modulo che l'intero programma. Quando tutti i moduli funzionano, occorre unirli tutti insieme e solo allora si può verificare l'intero sistema.

ESEMPIO DI PROGETTO SOFTWARE

Per illustrare le fasi di un progetto software, in questa sezione viene descritto un programma di controllo per semafori. Per prima cosa viene progettato il controllo fondamentale, poi vengono suggeriti alcuni miglioramenti. I LED delle porte d'uscita sono utilizzati per simulare le luci dei semafori.

Considerate un semplice incrocio tra due strade. Ciascun semaforo è costituito da una luce rossa, una gialla e una verde. Il controllo deve passare attraverso la seguente sequenza:

1. Semaforo A = rosso
2. Semaforo B = verde
3. Aspetta per la durata del verde
4. Cambia il semaforo B in giallo
5. Aspetta per la durata del giallo
6. Cambia il semaforo B in rosso
7. Cambia il semaforo A in verde
8. Aspetta per la durata del verde
9. Cambia il semaforo A in giallo
10. Aspetta per la durata del giallo
11. Ripeti tutta la sequenza (ritorna al punto 1).

Potreste programmare questa sequenza in modo diretto, ma non è la soluzione migliore, vista la lunghezza del programma. Da notare che il programma contiene due parti che vengono ripetute: una è la sequenza per il semaforo A (da verde a giallo a rosso), l'altra, identica, per il semaforo B. Si ottiene un maggiore efficienza se si usa un solo programma per entrambi i semafori.

La figura 13-1 mostra il diagramma di flusso per realizzare tale controllo. La routine sequenza gestisce entrambi i semafori. Il programma principale deve comunicare alla routine sequenza quale sia il semaforo da cambiare. Questo viene effettuato utilizzando un registro definito come «flag di semaforo». Prima di chiamare la routine sequenza il programma principale predispone questo registro in modo che la routine ne possa verificare il contenuto. Se il registro contiene zero, viene inserito nella sequenza il semaforo A. Se il registro non contiene zero viene gestito il semaforo B. L'operazione di passare delle informazioni da un programma chiamante ad una subroutine è detta *trasferimento dei parametri* (parameter passing).

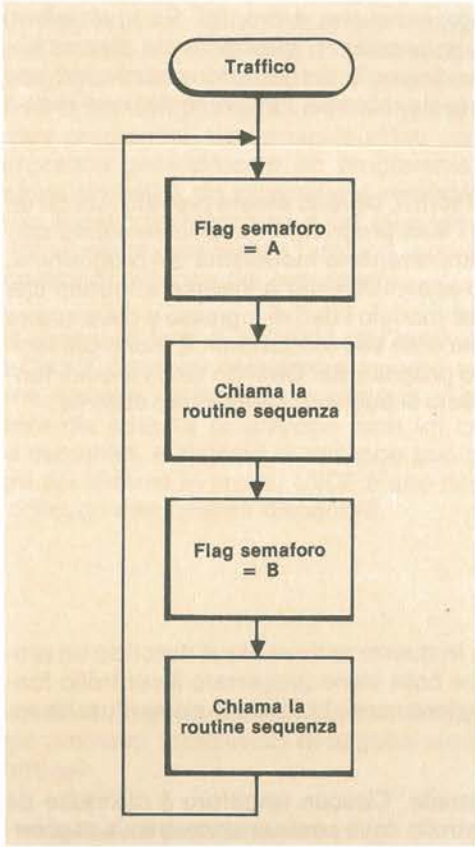


Figura 13-1a. Programma principale per il controllo dei semafori

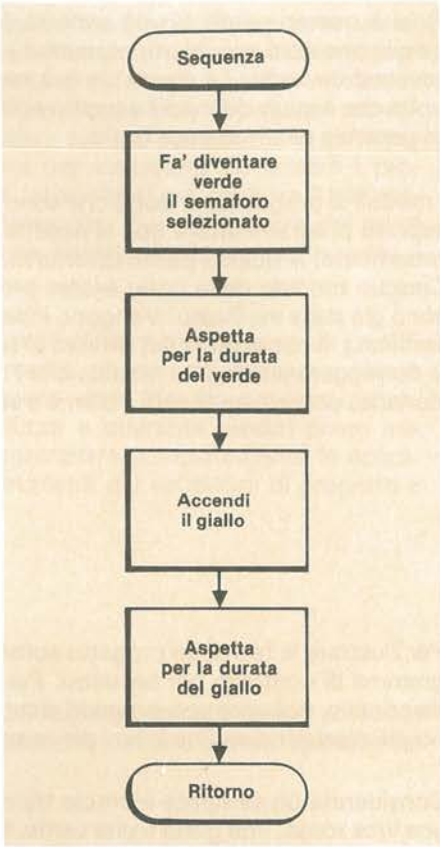


Figura 13-1b. Routine sequenza

È così definita l'architettura di base (struttura funzionale) del controllo per semafori. Essendo tutti i dettagli trattati dalla routine sequenza, il programma principale sarà molto semplice. Un programma ben strutturato avrà in genere un programma molto semplice.

LA ROUTINE SEQUENZA

Se il flag di semaforo è a zero, la routine sequenza pone, per prima cosa, il semaforo A verde e quello B rosso. Successivamente il semaforo A diviene giallo mentre B rimane rosso. Viceversa se il flag di semaforo è a uno, diventa rosso il semaforo A mentre il semaforo B diviene verde e poi giallo. Notate che non è necessario far diventare rosso il semaforo alla fine della sequenza. Questo viene fatto quando, all'inizio del prossimo ciclo, l'altro semaforo viene posto verde.

LA ROUTINE CAMBIO

Poiché la sequenza di operazioni è la stessa indipendentemente dal semaforo che è stato selezionato, il flag di semaforo non viene per nulla utilizzato dalla routine sequenza. La sola routine di uscita (che descriveremo tra breve) verifica lo stato di questo flag.

Come mostrato in Figura 13-2, tutte le luci dei semafori (i LED della porta di uscita) sono connesse ad un'unica porta di uscita. Due bit non sono utilizzati, i due LED che controllano sono perciò sempre spenti. La tabella 13-1 illustra le configurazioni di bit necessarie per simulare i semafori. Da notare che la differenza fra le prime due configurazioni (semaforo B rosso) e le seconde due (semaforo A rosso) consiste nel fatto che sono scambiati i quattro bit di sinistra con i quattro di destra. Questo non sorprende dal momento che i due semafori operano allo stesso modo e ciascuno dei due utilizza metà degli otto bit della porta di uscita.

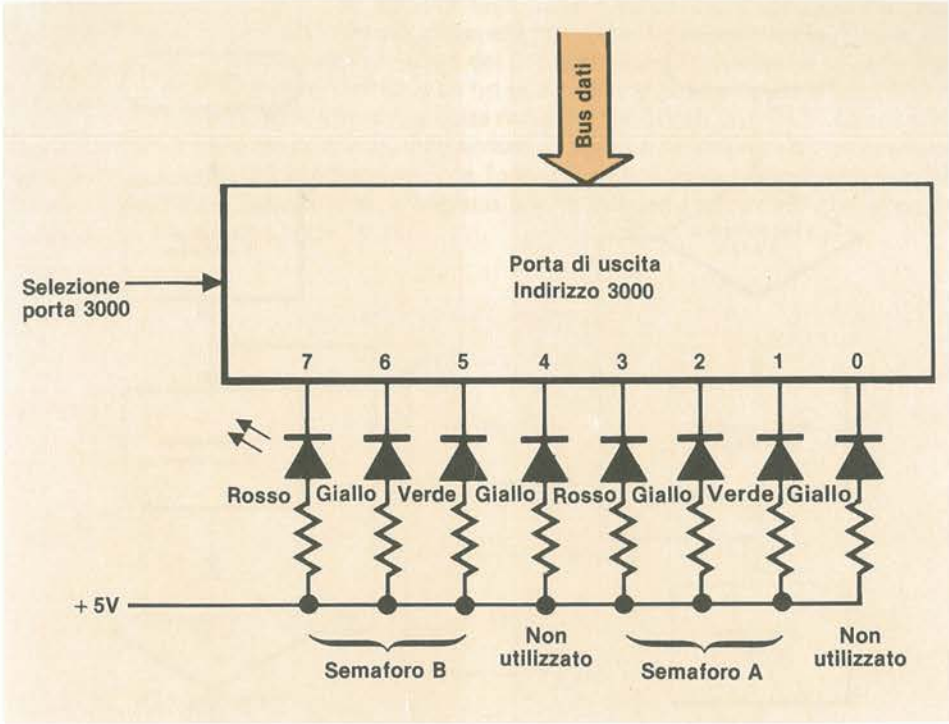


Figura 13-2. Collegamenti dei LED per simulare i semafori

	B				A				
	ROSSO	GIALLO	VERDE		ROSSO	GIALLO	VERDE		Hex
A Verde, B, Rosso	0	1	1	1	1	1	0	1	7D
A Giallo, B Rosso	0	1	1	1	1	0	1	1	7B
B Verde, A Rosso	1	1	0	1	0	1	1	1	D7
B Giallo, A Rosso	1	0	1	1	0	1	1	1	B7

Tabella 13-1. Configurazioni dei bit per simulare i semafori

Detto questo, può essere spiegata la logica delle routine sequenza e cambio. La routine sequenza tratta sempre la sequenza relativa al semaforo A (ovvero tratta sempre i quattro bit di destra). Essa richiama la routine cambio per realizzare l'operazione di uscita vera e propria. La Figura 13-3 riporta il diagramma di flusso della routine cambio. Se il flag di semaforo è a uno (il che significa che si tratta del semaforo B) la routine cambio commuta la metà destra e sinistra del dato prima di inviarla in uscita, eseguendo quattro istruzioni di rotazione.

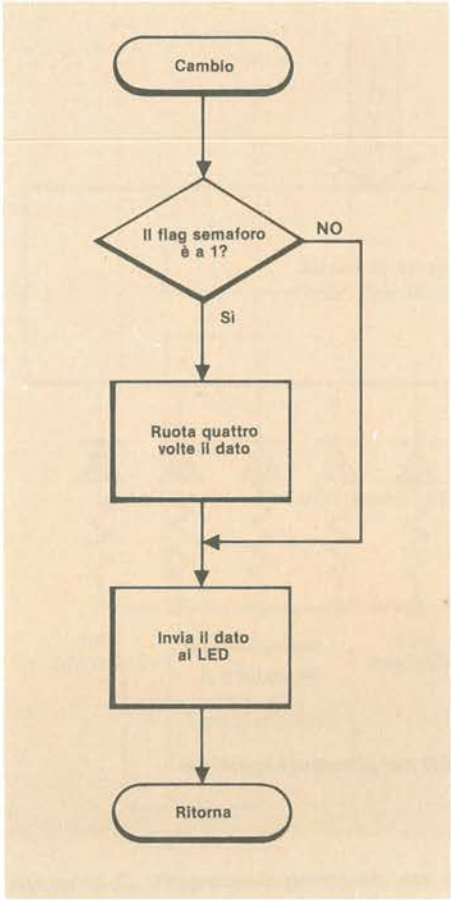


Figura 13-3. Routine di cambio semaforo (CHNG)

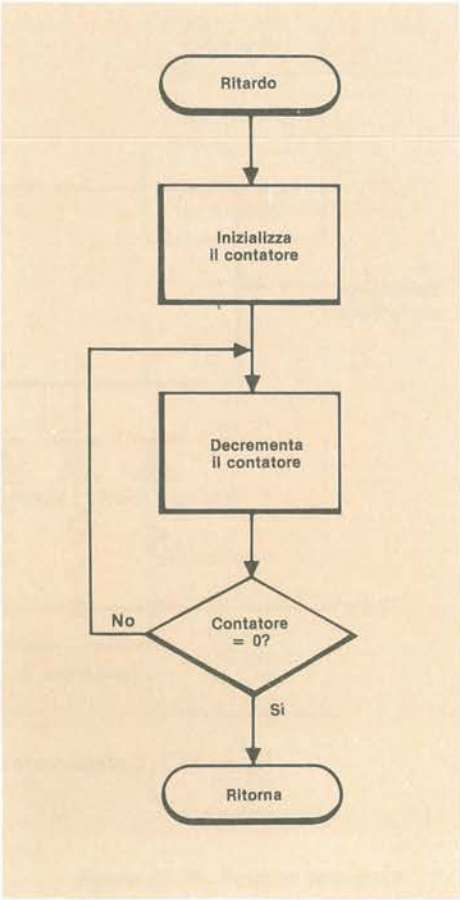


Figura 13-4. Routine di ritardo

ROUTINE DI RITARDO

Il sistema è ora progettato, manca solo la routine di ritardo (delay). Occorre una routine che fornisca un ritardo variabile da alcuni secondi (durata del giallo) a uno o due minuti (durata del verde).

In Figura 13-4 è mostrato un modo per ottenere dei tempi di ritardo. Un registro, che opera come contatore, viene inizializzato ad un determinato valore e poi decrementato finché raggiunge lo zero. Per decrementare, confrontare il registro e saltare all'inizio sono necessari circa sette microsecondi. Perciò, ritardi da sette μs fino a $7 \times 255 = 1785$ μs (1,785 ms) si possono ottenere inizializzando il contatore a valori compresi tra 1 e 255. Per ottenere un ritardo variabile, la routine di ritardo è scritta in modo che il valore di inizializzazione non sia definito dalla routine di ritardo stessa, ma che tale valore venga generato dal programma chiamante. Questo è un altro esempio di trasferimento di parametri.

Il programma di ritardo per il controllo dei semafori funziona sul principio appena indicato. Tuttavia, con la routine di Figura 13-4, esiste un problema. Essa ha un ritardo massimo inferiore a due millisecondi, mentre al semaforo occorrono ritardi di alcune decine di secondi. La soluzione consiste nell'estendere pari pari questa tecnica. Viene utilizzato un registro addizionale per estendere il valore di conteggio fino a 65.536. I dettagli di questa estensione vengono presentati più oltre in questa stessa lezione.

SCHEMI DI STRUTTURA

Uno schema di struttura è utile per mostrare come è suddiviso un programma. Esso mostra come e quali routine siano chiamate dai diversi programmi. In Figura 13-5 viene riportato uno schema di struttura del controllo semafori. La routine principale è chiamata TRAF ed è rappresentata da un rettangolo in cima allo schema. TRAF richiama la subroutine SEQ, che a sua volta richiama sia CHNG che DLY. Occorre notare che questo schema non è un diagramma di flusso e quindi non dà informazioni circa il flusso reale del programma, ma fornisce invece informazioni riguardo alla struttura logica del programma. Vengono anche indicati i parametri che vengono trasferiti da una routine ad un'altra.

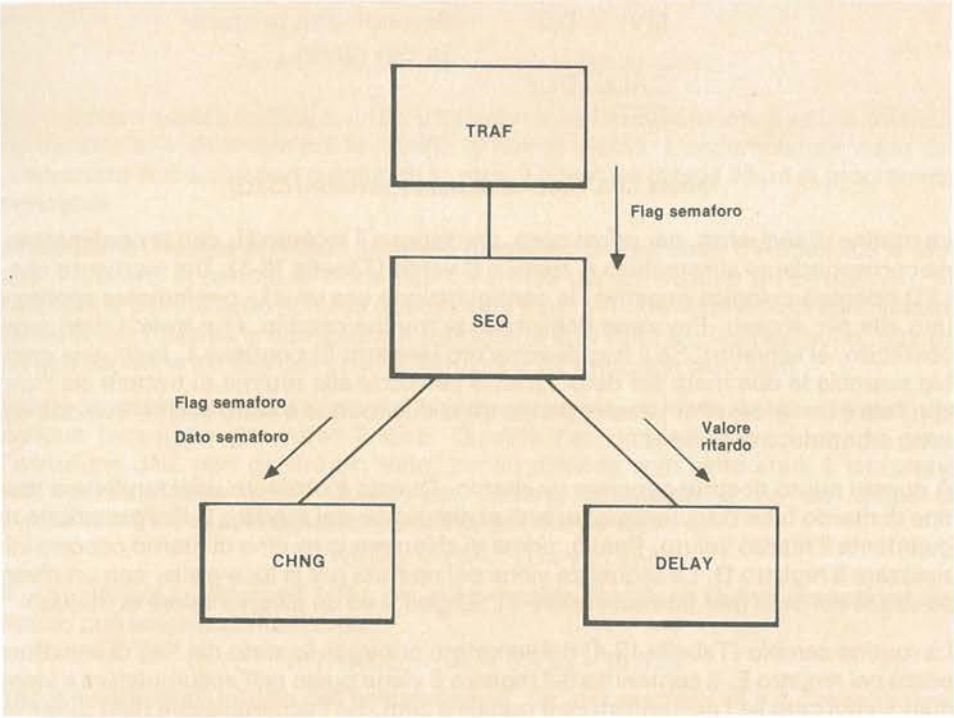


Figura 13-5. Schema funzionale del controllore di semafori

PROGRAMMI DI CONTROLLO

A questo punto abbiamo completato le specifiche e i diagrammi di flusso per ciascuno dei programmi richiesti. Per poter scrivere i programmi veri e propri, è necessario scegliere i diversi registri da associare a ogni funzione. Il registro E è utilizzato come flag di semaforo (la scelta è arbitraria). Come mostrato in Tabella 13-2 il programma principale è molto semplice.

```
TRAF: MVI  E,0    ;Sequenza semaforo A
      CALL SEQ
      MVI  E,1    ;Sequenza semaforo B
      CALL SEQ
      JMP  TRAF
```

Tabella 13-2. Programma principale per il controllo semafori (TRAF)

Questo programma segue da vicino il diagramma di flusso presentato precedentemente. In Tabella 13-3 è mostrata la routine di sequenza (SEQ).

```
SEQ:  MVI   H,7D    ;Accendi il verde
      CALL  CHNG
      MVI   D,6     ;Attendi per la dura-
                     ta del verde
      CALL  DLY
      MVI   H,7B    ;Accendi il giallo
      CALL  CHNG
      MVI   D,2     ;Attendi per la dura-
                     ta del giallo
      CALL  DLY
      RET
```

Tabella 13-3. Routine sequenza semaforo (SEQ)

La routine di sequenza, per prima cosa, predispone il registro H, con la configurazione corrispondente al semaforo A rosso e B verde (Tabella 13-1). Dal momento che i LED operano in logica negativa, la configurazione usa un «1» per indicare spento e uno «0» per acceso. Poi viene richiamata la routine cambio, che invia il dato, così costituito, ai semafori. Se il flag di semaforo (registro E) contiene 1, la routine cambio scambia le due metà del dato. Questo permette alla routine di trattare sia l'uno che l'altro semaforo. Per passare il dato tra le due routine è stato scelto, in modo del tutto arbitrario, il registro H.

A questo punto occorre generare un ritardo. Questo è ottenuto utilizzando una routine di ritardo (che discuteremo tra breve) che riceve nel registro D l'informazione riguardante il ritardo voluto. Perciò, prima di chiamare la routine di ritardo occorre inizializzare il registro D. La sequenza viene poi ripetuta per la luce gialla, con un diverso valore del dato (per fare accendere il LED giallo) ed un diverso valore di ritardo.

La routine cambio (Tabella 13-4) del semaforo controlla lo stato del flag di semaforo posto nel registro E. Il contenuto del registro E viene posto nell'accumulatore e viene quindi verificato se l'accumulatore è uguale a zero. Se l'accumulatore (flag di semaforo) è zero il flag di zero della CPU verrà posto ad uno. Sullo stato di flag viene poi effettuato un test attraverso l'istruzione JZ. Notate che l'istruzione MOV A,H posta tra le operazioni di confronto con zero (CPI 0) e di test (JZ) non interferisce con la operazione di test stessa, perché non altera i flag. L'istruzione JZ effettua un test sullo stato del flag di zero e non sul contenuto dell'accumulatore. Se il flag di semaforo non è posto a 1, il programma salta al passo di «uscita». Se invece il flag di semaforo è posto a 1 sul dato vengono effettuate quattro operazioni di rotazione onde scambiare la parte destra con quella sinistra dello stesso. Ogni operazione di rotazione sposta il dato di una posizione per volta (un bit).

```
CHNG: MOV  A,E      ;Il flag semaforo è = 1?
      CPI  0        ;Confronta il flag semaforo con zero
      MOV  A,H      ;Trasferisci il dato del semaforo
                     nell'accumulatore
      JZ   OUTP     ;Uguale a zero, invia il dato diretta-
                     mente in uscita
      RLC           ;Uguale a 1, ruota il dato
      RLC
      RLC
      RLC
OUTP: STA  3000     ;Invia il dato al semaforo
      RET
```

Tabella 13-4. Routine cambio semaforo (CHNG)

ROUTINE DI RITARDO

La routine di ritardo per il controllo dei semafori è leggermente complicata dalla lunghezza del tempo di ritardo richiesto. Prima di spiegare come sia stata realizzata, è utile dare alcuni esempi di routine di ritardo. La Tabella 13-5 mostra una semplice routine di ritardo, comunemente usata, che segue il diagramma di flusso di Figura 13-4.

```
DELAY: DCRA        ;Decrementa il contatore     4 stati
       JNZ DELAY   ;Ripeti finché non arriva    7/10 stati
                   a zero
       RET                                      10 stati
```

Tabella 13-5. Semplice routine di ritardo (Delay)

Per utilizzare questa routine è sufficiente inserire nell'accumulatore il valore del ritardo desiderato e chiamare poi la routine di ritardo stessa. L'accumulatore viene decrementato fino a che non raggiunge lo zero; il controllo ritorna allora al programma principale.

Analizziamo il tempo di ritardo generato da questo programma. L'Appendice B mostra il numero di periodi di clock (stati) richiesti per conseguire un'istruzione. Per l'istruzione DCR A sono richiesti quattro stati e per JNZ da sette a dieci stati. Il salto richiede dieci stati se la condizione è soddisfatta ed il salto è quindi eseguito, mentre ne richiede sette in caso contrario. L'istruzione RET richiede dieci stati.

Perciò la routine, ad eccezione dell'ultimo passaggio, richiede $4+10=14$ stati per ciascun passaggio attraverso il loop. Quando l'accumulatore raggiunge lo zero, l'istruzione JNZ non genera un salto, perciò richiede solo sette stati. L'istruzione RET richiede dieci stati. Quindi per l'ultimo passaggio si ha un totale di $4+7+10=21$. Per cui l'equazione che indica il ritardo totale è:

$$\text{RITARDO} = (A - 1)14 + 21 \text{ stati}$$

Il μLab utilizza un clock a 2 MHz, per cui uno stato dura 5 μs. Quindi l'equazione del ritardo può essere riscritta come:

$$\text{RITARDO} = (A - 1)7\,\mu s + 10{,}5\,\mu s$$

«A» è il contenuto iniziale dell'accumulatore. Se, per esempio, $A=1$ il ritardo sarà: $(1-1)7+10{,}5 = 10{,}5\,\mu s$. Da notare che il primo termine diviene zero perché il loop non viene mai eseguito; infatti la prima volta che si esegue il loop, l'accumulatore diviene zero, per cui il salto condizionato non viene mai effettuato.

Il ritardo massimo viene ottenuto ponendo l'accumulatore uguale a zero. La prima volta che si esegue il loop, l'accumulatore viene decrementato e posto a 255. Dal momento che sul contenuto dell'accumulatore è effettuato un test solo dopo che questo è già stato decrementato, è come se nell'equazione per il calcolo del ritardo avessimo inserito $A=256$. Per cui il ritardo massimo vale $(256-1)7+10{,}5 = 1.795$ μs. Questo valore è inferiore a due millesimi di secondo e in verità non è molto utile come intervallo di ritardo per il controllo dei semafori.

UTILIZZO DI COPPIE DI REGISTRI

L'8085 ha un gruppo di istruzioni che operano sulle coppie di registri invece che sul singolo registro. Anche se queste istruzioni non sono strettamente necessarie sono spesso utili. La routine per generare un ritardo lungo (Tabella 13-6) ci dà un buon esempio di come utilizzare queste istruzioni.

```
DELAY: DCX B       ;Decrementa il contatore
       MOV A,B     ;Verifica se è zero
       ORA C
       JNZ DELAY   ;Ripeti finché non arriva a zero
       RET
```

Tabella 13-6. Routine di ritardo che fa uso di una coppia di registri

L'istruzione DCX decrementa una coppia di registri, trattandoli come se contenessero un unico numero di sedici bit. Questo permette di contare fino a $21^{16} =$ 65.536. DCX B specifica che sono i registri B e C che devono essere decrementati come coppia. In un'istruzione che opera su coppie di registri, «B» fa riferimento alla coppia B e C. Se esiste un riporto dal registro C, esso viene così automaticamente sottratto dal registro B.

Sfortunatamente l'istruzione DCX non altera il flag di zero. Per alterare il flag di zero, occorre trasferire nell'accumulatore uno dei byte della coppia di registri di conteggio ed eseguire l'OR con l'altro byte. Il risultato è zero solo se entrambi i byte sono zero. A questo punto anche il flag di zero è posto a uno e può essere utilizzata in tal modo l'istruzione JNZ per controllare quando il conteggio passa per zero.

L'analisi dei tempi di questa routine è molto simile a quella già fatta precedentemente. L'istruzione DCX B richiede sei stati, la MOV A,B e ORA C ne richiedono quattro ciascuna, e le altre istruzioni sono le medesime viste nell'esempio precedente. L'equazione del ritardo diviene perciò:

$$\begin{aligned} \text{RITARDO} &= (N-1)24+31 \text{ stati} \\ &= (N-1)12\,\mu s+15{,}5\,\mu s \end{aligned}$$

dove «N» è il numero binario contenuto nei registri B e C. Il ritardo massimo è: $(65.536-1)12\,\mu s+15{,}5\,\mu s = 0{,}786$ sec. Questo valore è più vicino al valore richiesto per il controllo semafori, ma non è ancora sufficiente. Tuttavia, questa routine può ora essere usata per generare un ritardo ancora più lungo se combinata con una semplice routine di conteggio. In Figura 13-6 è mostrato il diagramma di flusso. Per generare un tempo di ritardo più lungo, la routine di ritardo di Tabella 13-5 è usata con il valore di ritardo posto sempre a zero (per ottenere il ritardo massimo di 0,786 secondi). Questo può essere fatto per un massimo di 256 volte in modo tale che il ritardo massimo ottenibile sarà di $256 \times 0{,}786$ secondi $= 201$ secondi, valore questo adeguato al controllo per semafori.

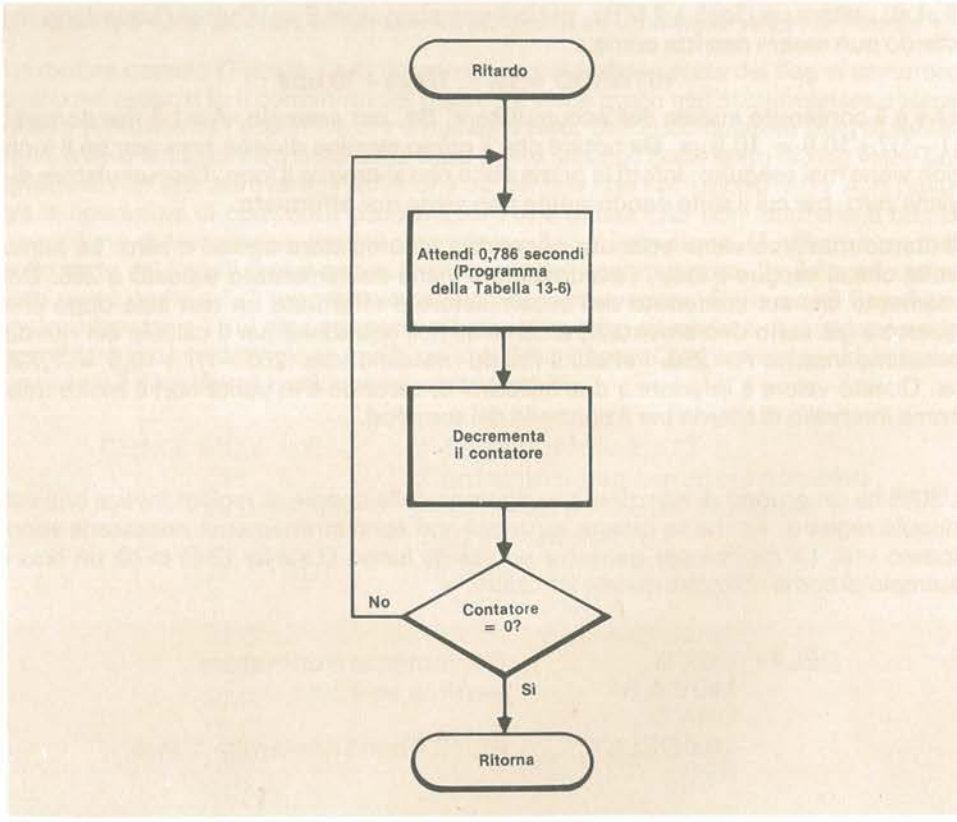


Figura 13-6. Routine per un ritardo lungo

La Tabella 13-7 riporta il listing del programma. Il ritardo di 0,786 secondi è realizzato con lo stesso programma già indicato. I registri B e C sono caricati con zero, in modo da ottenere il ritardo massimo. Questo si ottiene utilizzando l'istruzione LXI (*Load Register Pair Immediate*, carica la coppia di registri in modo immediato). Tale istruzione è simile alla istruzione MVI solo che in questo caso sono caricati entrambi i registri in una sola volta. Il codice operativo dell'istruzione LXI è seguito da due byte di dato (modo immediato). Il primo byte del dato è caricato nel registro C, mentre il secondo nel registro B.

Il registro D viene utilizzato per il conteggio del ritardo principale. Notate che questo programma possiede due loop. Uno, più interno, genera un ritardo base di 0,786 secondi e l'altro, più esterno, genera dei multipli di questo ritardo base. Strutture di loop così fatte sono dette *nidificate*.

La routine di ritardo è comunemente utilizzata in diversi tipi di applicazioni con microprocessori. Sono comunemente usate per realizzare oscillatori «programmati» e generatori di impulsi, per ovviare al problema del rimbalzo dei tasti della tastiera (di questo discuteremo nella prossima lezione) e in molte altre applicazioni.

```
DELAY: LXI   B,0     ; Inizializza B per il loop interno di ritardo
LOOP:  DCX   B       ; Loop interno di 0,786 secondi di ritardo
       MOV   A,B
       ORA   C
       JNZ   LOOP    ; Decrementa il contatore principale
       DCR   D
       JNZ   DELAY   ; e il loop a meno che sia zero
       RET
```

Tabella 13-7. Routine di ritardo per il controllo di un semaforo

TECNICHE DI COLLAUDO

È molto più facile isolare eventuali problemi ed errori, se si collaudano le routine separatamente, una per volta. La procedura generalmente seguita, prima di far girare la routine, è quella di preparare il contenuto dei registri, dai quali la routine preleva i parametri di ingresso, introducendovi da tastiera alcuni valori tipici e significativi. Spesso vengono utilizzati dei breakpoint (punti di fermata) per fermare il programma in certi punti interessati. In alternativa si può fare girare il programma passo passo (single step). Vengono poi esaminate le uscite della routine in modo da verificarne la correttezza: tutto questo può portare a dovere esaminare delle locazioni di memoria, i registri o gli stati di una porta di uscita.

È importante l'ordine in cui le routine vengono provate. Se per esempio viene provata una routine che richiama una routine non ancora verificata non è possibile conoscere quale routine sia la causa di un eventuale malfunzionamento. Per prima cosa occorre perciò provare le routine che non ne richiamano altre. Lo schema di struttura (Figura 13-5) identifica convenientemente queste routine (si trovano nella parte bassa dello schema). Successivamente dovrebbero essere provate le routine che chiamano solamente routine già verificate corrette. Queste routine sono quelle che si trovano al livello successivo nello schema di struttura.

Gli esperimenti seguenti esemplificano queste prove e il programma di controllo per i semafori. L'intero programma è presentato, con il relativo codice macchina, nella tabella 13-8. Al programma principale ed alle routine di sequenza sono state aggiunte delle istruzioni di NOP, in modo che sia più facile in un secondo tempo effettuare delle modifiche che introducano nuove prestazioni. È buona pratica aggiungere questi NOP in ogni programma. Per provare la vostra bravura nel codificare, coprite il codice macchina e cercate di tradurre voi stessi il programma. Controllate poi se il codice da voi scritto coincide con quello mostrato in Tabella 13-8.

PROGRAMMA PRINCIPALE

Indirizzo	Contenuto	Label	Istruzione		Commenti
0810	1E	TRAF:	MVI	E,0	;Sequenza semaforo A
0811	00			1E	
0812	CD		CALL	SEQ	
0813	30				
0814	08				
0815	00		NOP		
0816	00		NOP		
0817	00		NOP		
0818	00		NOP		
0819	1E		MVI	E,1	;Sequenza semaforo B
081A	01				
081B	CD		CALL	SEQ	
081C	30				
081D	08				
081E	C3		JMP	TRAF	
081F	10				
0820	08				

ROUTINE DI SEQUENZA

Indirizzo	Contenuto	Label	Istruzione		Commenti
0830	26	SEQ:	MVI	H,7D	;Accendi il verde
0831	7D				
0832	CD		CALL	CHNG	
0833	55				
0834	08				
0835	16		MVI	D,6	;Attendi per la durata del verde
0836	06				
0837	CD		CALL	DELAY	
0838	70				
0839	08				
083A	26		MVI	H,7B	;Accendi il giallo
083B	7B				
083C	CD		CALL	CHNG	
083D	55				
083E	08				
083F	16		MVI	D,2	;Attendi per la durata del giallo
0840	02				
0841	CD		CALL	DELAY	
0842	70				
0843	08				
0844	C9		RET		

Tabella 13-8. Programma per il controllore dei semafori

ROUTINE CAMBIO

Indirizzo	Contenuto	Label	Istruzione		Commenti
0855	7B	CHNG:	MOV	A,E	;Il flag semaforo è a 1?
0856	FE		CPI	0	
0857	00				
0858	7C		MOV	A,H	
0859	CA		JZ	OUTP	;Il flag non è a 1, invia il dato
085A	60				direttamente in uscita
085B	08				
085C	07		RLC		;il flag è a uno, ruota il dato
085D	07		RLC		
085E	07		RLC		
085F	07		RLC		
0860	32	OUTP:	STA	3000	;Invia il dato al semaforo
0861	00				
0862	30				
0863	C9		RET		

ROUTINE RITARDO (DELAY)

Indirizzo	Contenuto	Label	Istruzione		Commenti
0870	01	DELAY:	LXI	B,0	;Inizializza B per il loop inter-
0871	00				no di ritardo
0872	00				
0873	0B	LOOP:	DCX	B	;Loop interno di 0,786 secon-
0874	78		MOV	A,B	di di ritardo
0875	B1		ORA	C	
0876	C2		JNZ	LOOP	
0877	73				
0878	08				
0879	15		DCR	D	;Loop principale
087A	C2		JNZ	DELAY	
087B	70				
087C	08				
087D	C9		RET		

Tabella 13-8. Programma per il controllore dei semafori (segue)

ESPERIMENTO 13-1

Collaudo del programma di controllo per semafori

INTRODUZIONE

In questo esperimento inserite e collaudate una per volta le routine del programma di controllo per semafori. Il programma sarà collaudato quando tutte le routine saranno correttamente funzionanti. Sono ripetutamente usate le caratteristiche di breakpoint e di accesso ai registri del μLab.

PROCEDIMENTO

A) Da tastiera, caricate le quattro routine della Tabella 13-8. Notate che tra una routine e l'altra è stato lasciato dello spazio, utilizzabile per eventuali modifiche delle stesse.

B) Verificate che le routine siano state memorizzate correttamente.

C) Per prima cosa provate la routine di ritardo. Per collaudare tale routine è opportuno fermarla quando è terminata. Dal momento che non è stata chiamata da un'altra routine, non dovete far eseguire l'istruzione di ritorno. Perciò, sostituite l'istruzione RET, alla fine delle routine di ritardo, con un breakpoint (RST 1, codice operativo CF).

D) Ponete nel registro D il valore 5. Per accedere al registro D premete [FETCH REG] e [STORE /INCR] per quattro volte. Poi premete [5] [STORE /INCR]. Questo vi permette di predisporre a cinque il valore di ritardo.

E) Fate girare la routine di ritardo (indirizzo 0870). Il display dovrebbe rimanere spento per circa quattro secondi (5 × 0,786) e dovrebbe poi mostrare l'indirizzo che segue il breakpoint (087E). Se non funziona, confrontate il contenuto della memoria con la Tabella 13-8. Controllate che l'istruzione RET sia stata effettivamente sostituita con RST 1 e verificate se avete eseguito correttamente il punto D.

F) Ponete un altro valore nel registro D e fate partire la routine. Calcolate il ritardo che dovreste così avere generato e controllate quello reale per vedere se corrisponde al vostro calcolo.

G) Se i tempi calcolati e misurati sono uguali, la routine funziona bene. Sostituite nuovamente RST 1 con RET.
La routine di ritardo è stata provata e potete ora provare la routine cambio CHNG.

H) La routine CHNG si attende in ingresso due valori: il flag di semaforo nel registro E, e il dato per il semaforo nel registro H. Ponete il registro E a zero e quello H a 7D (0111 1101); così facendo l'esecuzione della routine porterà il semaforo di sinistra ad essere rosso e quello di destra verde.

I) Sostituite l'istruzione RET alla fine della routine (indirizzo 0863) con un breakpoint.

J) Fate eseguire la routine CHNG (partendo con l'indirizzo 0855). I semafori (LED) dovrebbero essere predisposti come indicato al punto H.

K) Ponete a 1 il registro E. Poiché il registro E è il flag di semaforo, in questo modo invertite i due semafori.

L) Fate eseguire la routine CHNG. Il semaforo di sinistra dovrebbe essere ora verde e l'altro rosso.

M) Inserite nel registro H un nuovo dato, in modo da avere un semaforo rosso e l'altro giallo (riferitevi alla Tabella 13-1).

N) Fate eseguire la routine CHNG e verificate che i semafori siano accesi correttamente.

O) Se i semafori operano come previsto la routine è corretta. Altrimenti controllate che il programma memorizzato sia esatto e che la procedura di prova (iniziata al punto H) sia stata realizzata correttamente.

P) Sostituite ancora il breakpoint all'indirizzo 0863 con l'istruzione RET. La routine CHNG è stata così verificata, per cui a questo punto siete pronti a provare la routine SEQ.

Q) L'unico ingresso che occorre fornire alla routine sequenza è il flag di semaforo (registro E). Ponete il registro E a zero. Questo fa sì che sia selezionato il semaforo di destra.

R) Sostituite l'istruzione RET alla fine della routine (indirizzo 0844) con un breakpoint.

S) Fate eseguire la routine SEQ (indirizzo 0830). Un semaforo dovrebbe passare dal verde al giallo, ma non dovrebbe diventare rosso. Questo si verificherà nel passaggio successivo, quando l'altro semaforo diventerà verde; tale parte del programma non è ancora stata eseguita.

T) Ponete a 1 il registro E in modo da collaudare anche l'altro semaforo.

U) Fate eseguire ancora la routine SEQ. L'altro semaforo dovrebbe passare da verde a giallo.

V) La routine è corretta se viene eseguita la sequenza appena descritta. Sostituite ancora il breakpoint all'indirizzo 0844 con RET.

A questo punto tutti i moduli sono stati provati e quindi siete in grado di verificare l'intero sistema.

W) Fate eseguire il programma principale (indirizzo 0810). Ogni semaforo dovrebbe seguire la sequenza periodica di commutazione. Se questo non avviene controllate che tutti i breakpoint siano stati sostituiti con le istruzioni RET e che il programma principale sia stato memorizzato correttamente.

X) I valori di ritardo sono stati inseriti in modo non realistico, ma solo per far eseguire velocemente il programma. Provate ad introdurre dei valori diversi.

RIASSUNTO

Con questo esperimento avete collaudato e fatto girare il programma per il controllo dei semafori. Ciascuna routine è stata provata singolarmente, partendo da quelle di livello più basso (quelle che non ne richiamano altre). Eseguendo il collaudo in questo modo si rende minimo il numero possibile di sorgenti di errore e gli inconvenienti possono essere evidenziati più facilmente.

COME MIGLIORARE IL CONTROLLO PER SEMAFORI

Ci sono molti modi per migliorare questo controllo per semafori. Uno dei fattori da considerare è che le due strade possono avere condizioni di traffico diverse, per cui il verde dovrebbe avere durata diversa per ciascuna di esse.

Per aggiungere questa modifica, la routine sequenza, oltre al flag di semaforo, deve accettare un altro parametro: il tempo di durata del verde. Questa modifica è abbastanza facile. Per prima cosa, occorre cancellare l'istruzione MVI D,6 dalla routine SEQ (questa istruzione definisce la durata del verde). Occorre poi aggiungere al programma principale altre istruzioni che definiscano il contenuto del registro D (durata del verde). In Tabella 13-9 è presentato il nuovo programma principale.

Indirizzo	Contenuto	Label	Istruzione	Commenti
080E	16	TRAF:	MVI D,6	;Stabilisci la durata del verde per A
080F	06			
0810	1E		MVI E,0	;Sequenza per il semaforo À
0811	00			
0812	CD		CALL SEQ	
0813	30			
0814	08			
0815	00		NOP	
0816	00		NOP	
0817	16		MVI D,10	;Stabilisci la durata del verde per B
0818	10			
0819	1E		MVI E,1	;Sequenza per il semaforo B
081A	01			
081B	CD		CALL SEQ	
081C	30			
081D	08			
081E	C3		JMP TRAF	
081F	0E			
0820	08			

Tabella 13-9. Programma principale per il controllo semafori con diverse durate del verde

ESPERIMENTO 13-2

Modifica del controllo per semafori

INTRODUZIONE

In questo esperimento viene modificato il controllo dei semafori, al fine di realizzare durate del verde differenti per ciascuno dei due semafori.

PROCEDIMENTO

A) Se il programma di controllo semafori della Tabella 13-8 non è più presente nella memoria del μLab, introducetelo da tastiera (tranne il programma principale).

B) Sostituite l'istruzione MVI D,6 della routine sequenza (indirizzi 0835 e 0836) con due NOP.

C) Codificate il nuovo programma principale (Tabella 13-9) e caricatelo in memoria.

D) Fate girare il programma di controllo semafori e cronometrate i tempi per vedere se sono diversi.

E) Per verificare che il programma funzioni come desiderato, variate il tempo di durata del verde di ciascun semaforo.

RIASSUNTO

Questo esperimento ha illustrato una semplice modifica del programma di controllo per semafori. La struttura modulare del programma ha reso di facile implementazione questa modifica.

ESERCIZIO DI PROGRAMMAZIONE N. 3 MODIFICHE DEL CONTROLLO SEMAFORI

Il programma per il controllo dei semafori può essere migliorato in molti altri modi, di cui qui di seguito riportiamo alcuni esempi:

1) Dare diversi tempi di giallo per ciascun semaforo.
2) Accendere per entrambi i semafori il rosso per un breve periodo invece di passare direttamente dal rosso al verde.
3) Utilizzare i due LED rimasti per inserire dei segnali di svolta a sinistra.
4) Utilizzare uno degli interruttori della porta di ingresso per simulare un sensore sulla strada e non permettere così che il semaforo cambi a meno che non ci siano delle macchine in attesa sulle corsie dove il semaforo è rosso.

Queste sono solo alcune delle possibilità. La prima richiede alcuni limitati cambiamenti al programma principale ed alla routine sequenza. La seconda può essere realizzata con alcune aggiunte alla routine sequenza e lo stesso vale per la terza possibilità. La quarta è un poco più complicata e la risoluzione viene lasciata a voi.

Modificate il programma, in modo da inserire una o più di queste nuove funzioni. Se pensate di aggiungerne più di una, procedete un passo alla volta, provando il programma dopo ogni modifica. Vi sarà più facile risolvere i problemi che potete incontrare. La soluzione di questi esercizi si trova in Appendice A.

Il primo passo da seguire nel progettare un programma è quello di definire il problema. Avendo fissato il modo generale di operare del programma, può essere progettata una sua soluzione. In particolare può essere vantaggioso dividere il programma in routine piccole e modulari. Il programma principale utilizza le routine e definisce così un flusso di operazioni senza dover trattare i dettagli delle stesse. I diagrammi di flusso delle routine e la loro traduzione in codice vengono realizzati solo quando la struttura generale è già stata progettata. Da ultimo, vengono caricate in memoria e provate le subroutine e il programma principale.

L'utilizzo di una struttura modulare comporta numerosi vantaggi. Per prima cosa, suddividere un problema in tanti piccoli problemi rende più facile la soluzione, dal momento che ogni problema può essere studiato separatamente dagli altri. I programmi ben strutturati sono più facili da scrivere, più facili da correggere, più facili da modificare e più facili da capire.

Le routine comunicano tra di loro passandosi dei parametri. I parametri usati dal controllo dei semafori in particolare sono: flag di semaforo, dato di semaforo, valore di ritardo.

Le routine di ritardo sono utilizzate per generare degli intervalli di tempo tra due eventi. Il tempo impiegato per eseguire un loop di programma può essere calcolato in modo esatto, poiché il tempo impiegato per eseguire una istruzione è fissato dalla frequenza del clock, che è a sua volta controllata mediante un cristallo. Eseguendo poi il loop un certo numero di volte, si può ottenere il tempo di ritardo desiderato.

DOMANDE

Lezione 13

1. Il primo passo da fare quando si progetta un programma è quello di definire _____.

2. Un buon modo per realizzare un programma complesso è quello di scrivere:
   a. un unico programma.
   b. una serie di programmi che vengono eseguiti in sequenza.
   c. un programma principale che richiama delle routine che eseguano ciascuna delle funzioni che deve essere effettuata.
   d. un programma principale e una subroutine.

3. La prima parte di programma che va verificata è:
   a. il programma principale.
   b. la routine di livello più alto.
   c. la routine di livello più basso.
   d. la routine più semplice.

4. Un parametro è:
   a. un valore passato da una routine ad un'altra.
   b. il contenuto del program counter.
   c. l'indirizzo di ritorno.
   d. il numero di volte che viene eseguito un programma.

LEZIONE 14

Controllo software delle periferiche

INTRODUZIONE

In questa lezione viene descritto il software per gestire la tastiera e il display del Microprocessor Lab. Vengono descritti, in due fasi, i programmi che permettono l'ingresso da tastiera e l'uscita verso il display. Per prima cosa vengono descritte le subroutine del monitor di controllo tastiera e display, in seguito viene presentato in dettaglio un programma che spiega il software richiesto per interfacciarsi alla tastiera ed al display. Dal momento che la maggior parte dei sistemi comprende una tastiera ed un display, i concetti descritti in questa lezione sono direttamente applicabili a molti dei sistemi basati su microprocessore.

TASTIERA

La tastiera è disposta come una matrice di tasti in cui ciascuna riga di tasti viene letta separatamente (l'hardware è descritto nella Lezione 9). Il dato, ricavato leggendo ogni riga, è convertito nel codice corrispondente al tasto premuto tramite una routine, memorizzata nella ROM del μLab, chiamata KIND (Key Input and Decode, Ingresso e decodifica del tasto). Questa, e molte altre routine, sono riportate nell'Appendice E. L'utilizzazione della routine è molto semplice. È sufficiente infatti eseguire la chiamata di tale routine; la stessa routine effettuerà un ritorno non appena sia stato premuto un tasto. Nell'accumulatore sarà contenuto il codice del tasto premuto. Nella Tabella 14-1 sono mostrati i codici corrispondenti ai tasti. Questa subroutine del monitor può essere utilizzata dai vostri programmi quando vogliate leggere la tastiera.

TASTO	CODICE
0	00
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
A	0A
B	0B
C	0C
D	0D
E	0E
F	0F
FETCH REG	80
DECR	81
FETCH ADRS	82
STORE/INCR	83
RUN	84
FETCH PC	85
INSTR STEP	86
HDWR STEP	F7

Tabella 14-1. Codici dei tasti per la routine KIND

ESPERIMENTO 14-1

Utilizzazione della routine di lettura tastiera

INTRODUZIONE

Viene utilizzata la routine KIND, presente all'interno del monitor, per leggere la tastiera. Il codice del tasto premuto viene confrontato con un certo valore e, se è stato premuto il tasto desiderato, viene emesso un suono di avvertimento (bip, in inglese beep).

PROCEDIMENTO

A) Codificate il programma di Tabella 14-2 e caricatelo in memoria. Questo programma illustra una semplice applicazione della tastiera. Vengono usate due subroutine del monitor: KIND (indirizzo 014B) e BEEP (indirizzo 0010).

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	_____	READ:	CALL KIND	;Lettura stato
0801	_____			
0802	_____			
0803	_____		CPI 07	;Confronta il codice tasto
0804	_____			
0805	_____		JNZ READ	
0806	_____			
0807	_____			
0808	_____		CALL BEEP	;Genera un bip se è il ta-
0809	_____			sto "7"
080A	_____			
080B	_____		JMP READ	
080C	_____			
080D	_____			

Tabella 14-2. Programma per leggere la tastiera e generare un bip se viene premuto il tasto «7»

La tastiera viene letta utilizzando la routine KIND. Da questa routine si esce, dopo che è stato premuto un tasto, il codice del tasto premuto essendo contenuto nell'accumulatore. L'istruzione CPI 07 pone a uno il flag di zero se l'accumulatore è uguale a sette. L'istruzione JNZ READ fa ritornare il programma all'inizio se il flag di zero non è a uno. Se il flag di zero è uguale a uno, questo significa che il tasto premuto era proprio il sette. L'istruzione JNZ non ha allora alcun effetto e viene quindi chiamata la routine BEEP. Il processo poi si ripete, in modo tale che viene generato un segnale acustico tutte le volte che è premuto il tasto «7».

B) Verificate che il programma sia memorizzato correttamente.

C) Fate eseguire il programma. Tenete presente che quando viene premuto il tasto [RUN] apparentemente non succede nulla. Il programma sta girando ma poiché la routine KIND provvede alla scansione del display mentre effettua la lettura della tastiera, il display rimane acceso. Il programma monitor non sta girando.

D) Premete [7]. L'altoparlante emetterà un bip. Ora provate con un tasto diverso da «7»: solamente il tasto 7 genera il segnale acustico.

E) Premete [RESET] per ridare il controllo al monitor. Modificate il programma perché possa rispondere a un tasto diverso (usate la Tabella 14-1).

F) Provate il programma modificato.

G) Premete [RESET] per riportare il controllo al monitor.

RIASSUNTO

La routine del monitor di lettura tastiera (KIND) è stata usata per leggere i dati impostati da tastiera. Ogniqualvolta si è premuto un particolare tasto, è stato generato un segnale acustico. Cambiando il valore in cui viene confrontato il codice del tasto si può rilevare un qualsiasi tasto. Si è potuto vedere che, con l'utilizzo della subroutine KIND del monitor, è particolarmente facile leggere la tastiera.

ESERCIZIO DI PROGRAMMAZIONE N. 4: ANTIFURTO ELETTRONICO

Scrivete un programma, simile a quello della Tabella 14-2, che generi un segnale acustico solo quando due tasti vengano premuti in un ordine ben preciso. Questo programma potrebbe essere il punto di partenza per la realizzazione di un antifurto elettronico. Iniziate disegnando il diagramma di flusso e scrivete poi il programma vero e proprio. Introducetelo nella memoria del μLab, provatelo ed eventualmente correggetelo.

SCANSIONE DELLA TASTIERA

Se è vero che la routine del monitor di lettura tasti rende facile l'uso della tastiera, è anche vero che non abbiamo avuto così molte opportunità per vedere cosa succede veramente nel corso del processo di lettura. Questo paragrafo descrive un programma che legge la tastiera senza utilizzare la subroutine del monitor, in modo da spiegare la tecnica che viene utilizzata per leggere la tastiera.
In Figura 14-1 appare lo schema dell'interfaccia tastiera. Come descritto nella Lezione 9, la tastiera viene «scandita» riga per riga. Per semplicità, considerate la lettura di una sola riga di tasti (per esempio i tasti «1», «2» e «3»).

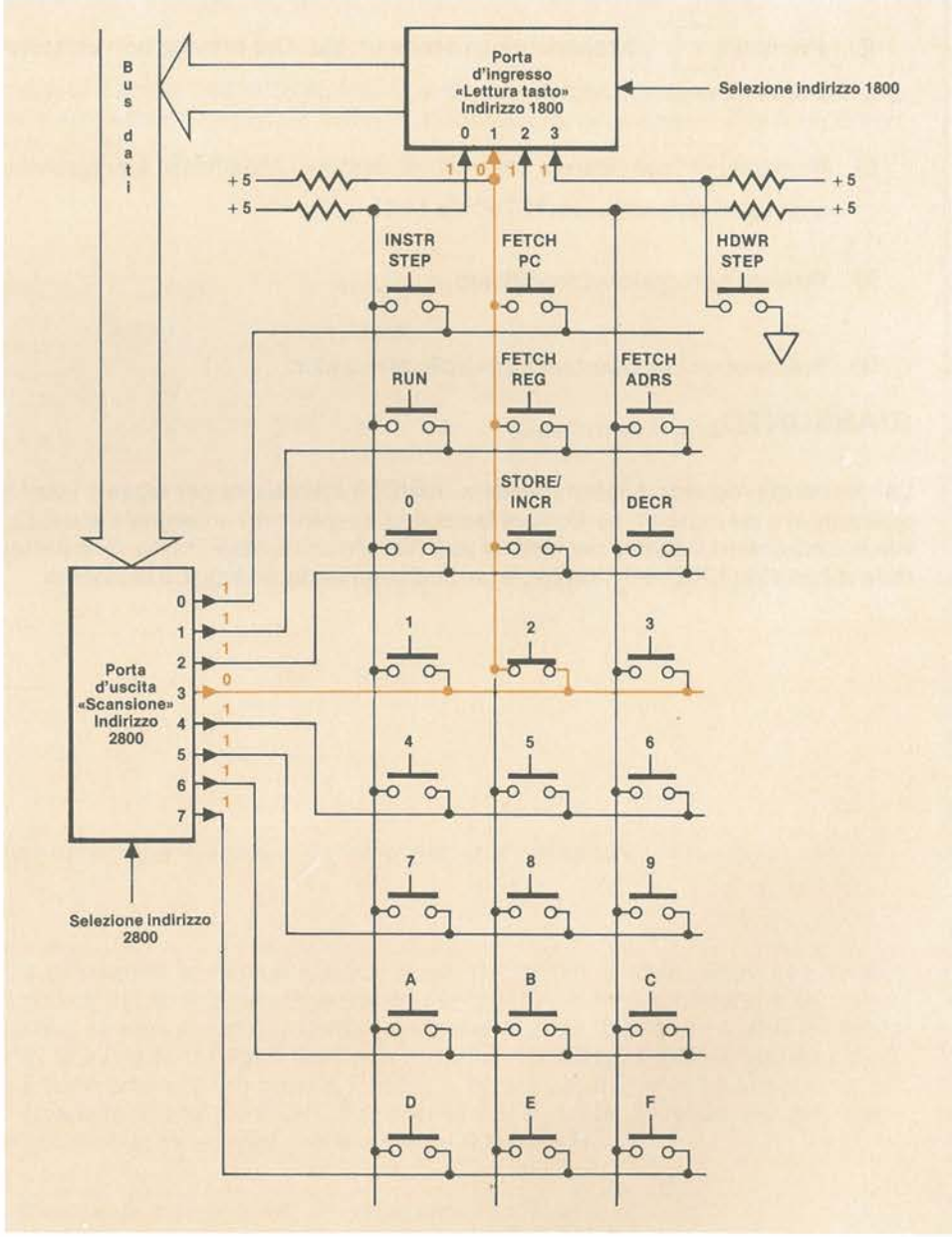


Figura 14-1. Interfaccia tastiera. È premuto il tasto 2.

La lettura di una riga di tasti è un'operazione composta da due momenti:

- Scrittura di un dato sulla porta di scansione, in modo da selezionare la riga desiderata.
- Lettura del dato di colonna dalla porta di lettura dei tasti.

Ogni bit della porta di scansione viene posto alto (1 logico); fa eccezione il bit che pilota la riga che si vuole leggere che viene posto basso. Perciò, per poter leggere i tasti 1, 2 e 3, viene scritto il dato 1111 0111 (F7 in esadecimale) nella porta di scansione (Figura 14-1). A questo punto, può essere letta la porta d'ingresso dei tasti, in modo da ottenere l'informazione di colonna. Se non vengono premuti dei tasti, i bit 0—3 sono tutti alti (uno), poiché sono richiamati alti dalle resistenze. Se viene premuto il tasto 2, diventa basso il bit 1, mentre, se viene premuto il tasto 3, diventa basso il bit 2 (Tabella 14-3). Dal momento che la porta di scansione seleziona una sola riga per volta, (tutte le altre sono «spente») nessun altro tasto può influenzare i dati letti dalla tastiera.

nessun tasto premuto	XXXX	X111
"1" tasto premuto	XXXX	X110
"2" tasto premuto	XXXX	X101
"3" tasto premuto	XXXX	X011

Tabella 14-3. Codici tasti. Le «X» nei cinque bit più significativi indicano che questi bit contengono dei valori sconosciuti. L'informazione che ci interessa è contenuta nei tre bit meno significativi

Notate che il valore letto dalla tastiera è un codice che identifica il tasto, ma non è il valore reale del tasto. Se, per esempio, viene premuto il tasto «2», l'accumulatore conterrà 05 hex (supponendo che i cinque bit più significativi siano tutti posti a zero). La routine KIND usata nel precedente esperimento scandisce e legge tutte le righe e traduce i valori letti nei corrispondenti valori dei tasti.

La Tabella 14-4 mostra un programma che effettua una operazione di lettura come quella appena descritta. Per prima cosa, viene predisposta la porta di scansione in modo da selezionare la riga desiderata, poi viene letto e posto nell'accumulatore il dato di colonna.

```
MVI A,F7     ;Definisci sulla porta scansione il valore
             1111 0111, in modo da selezionare una riga
STA 2800
LDA 1800     ;Leggi le colonne dei tasti
```

Tabella 14-4. Programma che legge una riga dalla tastiera

ESPERIMENTO 14-2

Scansione della tastiera

INTRODUZIONE

In questo esperimento viene letta la tastiera, avendo prima definito un certo valore sulla porta di scansione e leggendo poi la porta d'ingresso dei tasti. Viene così mostrato il principio base del programma di lettura tastiera del monitor, che è stato usato nel precedente esperimento. Il dato di colonna viene confrontato col valore atteso per un certo tasto; se il confronto dà un risultato positivo, viene generato un suono di avvertimento.

PROCEDIMENTO

A) Codificate ed introducete da tastiera il programma della Tabella 14-5. Questo programma, come già detto, legge una riga di tasti. Per portare a zero tutti i bit del dato, esclusi i tre meno significativi, il dato stesso viene posto in AND con una maschera (0000 0111). Sono infatti questi i tre bit che contengono il dato significativo proveniente dalla tastiera (Tabella 14-3). Il risultato viene confrontato con il valore 0000 0101, in modo da verificare se è stato effettivamente premuto il tasto «2». Se i valori sono uguali, viene generato un segnale acustico dalla routine BEEP del monitor.

Indirizzo	Contenuto	Label	Istruzione		Commenti
0800	_____		MVI	A,F7	;Definisci sulla porta di scan-
0801	_____				sione il valore 1111 0111
0802	_____		STA	2800	
0803	_____				
0804	_____				
0805	_____	READ:	LDA	1800	;Leggi le colonne
0806	_____				
0807	_____				
0808	_____		MVI	B, 07	;Tieni solo i tre LSB e ma-
0809	_____				schera tutti gli altri a zero
080A	_____		ANA	B	
080B	_____		CPI	05	;Il dato è 101 (Tasto "2")?
080C	_____				
080D	_____		JNZ	READ	;Se no continua a leggere
080E	_____				
080F	_____				
0810	_____		CALL	BEEP	;Se si genera un bip
0811	_____				
0812	_____				
0813	_____		JMP	READ	;Riprendi la lettura
0814	_____				
0815	_____				

Tabella 14-5. Programma che verifica quando è premuto il tasto «2»

B) Verificate che il programma sia memorizzato correttamente.

C) Fate partire il programma e provate a premere diversi tasti.

D) Premete [RESET] per ridare il controllo al monitor. Modificate il programma in modo da riconoscere il tasto «3» invece che il «2» (Tabella 14-3).

E) Fate partire il programma e verificate che esso operi come previsto. Se non riuscite a farlo funzionare, andate a vedere la soluzione che è nell'Appendice A.

RIASSUNTO

È stata letta la tastiera, definendo per prima cosa un valore sulla porta di scansione, in modo da selezionare la riga, e leggendo poi il dato di colonna. Il dato presentato sulla porta di scansione disabilita tutti i tasti tranne tre. Confrontando il dato letto da tastiera con le configurazioni di bit che sono generate quando viene premuto un tasto, è stato identificato quale tasto è stato premuto. Per riconoscere il tasto «3», la locazione 080C deve essere modificata in 03.

ANTIRIMBALZO

Quando si esegue la lettura dei dati provenienti da tastiera, occorre tenere in particolare considerazione il problema dell'antirimbalzo (Debounce). Ogni volta che vengono premuti, i tasti purtroppo non si chiudono in modo «pulito», ma «rimbalzano» come mostrato in Figura 14-2. La prima volta che i due contatti metallici si toccano, si verifica un rimbalzo che provoca la riapertura dell'interruttore. Immediatamente dopo i contatti si richiudono, per rimbalzare ancora un'altra volta. Lo stesso effetto si verifica anche quando si riapre l'interruttore cioè quando si rilascia il tasto. A seconda di come è costruito l'interruttore, i rimbalzi possono continuare per un tempo variabile da 1 a 50 ms.

Il rimbalzo dei contatti è importante, perché può far credere al microprocessore che un tasto sia stato premuto più volte, mentre, in effetti, era stato premuto una sola volta. Occorre quindi ignorare completamente le oscillazioni presenti all'inizio e alla fine di ogni impulso. Tutto questo può essere ottenuto aggiungendo alla porta d'ingresso dei circuiti di antirimbalzo o intervenendo da programma.

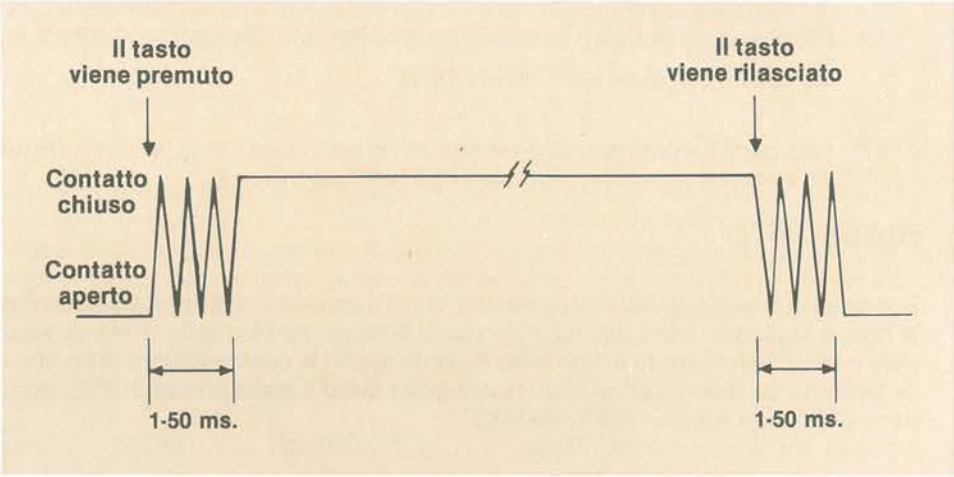


Figura 14-2. Rimbalzo di un interruttore

La Figura 14-3 mostra un modo per ovviare al problema del rimbalzo dei tasti della tastiera. Si esegue la scansione dei tasti utilizzando il procedimento già visto. Se non viene premuto nessun tasto, questa operazione viene ripetuta. Quando è sentito un tasto premuto, viene effettuata l'operazione indicata dal tasto stesso. Onde evitare di leggere il tasto mentre sta rimbalzando, la tastiera è quindi ignorata per 50 ms. Terminato tale periodo la scansione della tastiera continua ad essere eseguita (fino al rilascio del tasto). Quando il tasto viene finalmente rilasciato, si inserisce un altro ritardo di 50 ms, anche stavolta al fine di evitare la lettura della tastiera durante il periodo in cui il tasto oscilla.

IL DISPLAY

Il μLab utilizza un display a LED a sette segmenti, formato da sei cifre. Ad ogni istante è accesa una sola cifra: l'intero display viene illuminato scrivendo ripetutamente su tutte le cifre, una dopo l'altra. Questa tecnica è nota come *rinfresco* (Refresh) o *scansione* (Scan) del display. Ogni cifra rimane accesa solo per un sesto dell'intero tempo di scansione, ma, poiché tale tempo è molto breve, il display sembra sempre tutto acceso. Generalmente vengono pilotati in questo modo tutti i display a più cifre poiché tale soluzione semplifica lo hardware (come descritto nella Lezione 9).

Nel programma monitor del μLab esiste una subroutine chiamata DCD (Display Character Decoder, decodifica di caratteri per il display) che controlla il display. Per utilizzare questo programma, occorre memorizzare in sei locazioni di memoria (una per ogni cifra) le cifre che si vogliono visualizzare. Il programma legge dalla memoria tali cifre, le converte in codici adatti al display ed esegue infine il refresh ciclico dello stesso.

C'è un altro programma utile per gestire un display: si tratta della routine chiamata STDM (Store Display Message, memorizza il messaggio da visualizzare), che trasferisce semplicemente il messaggio (i caratteri cioè da visualizzare), dal vostro programma, alla posizione di memoria in cui la routine di gestione del display si aspetta di trovarlo. Utilizzando le due routine ora descritte, risulta particolarmente facile controllare un display.

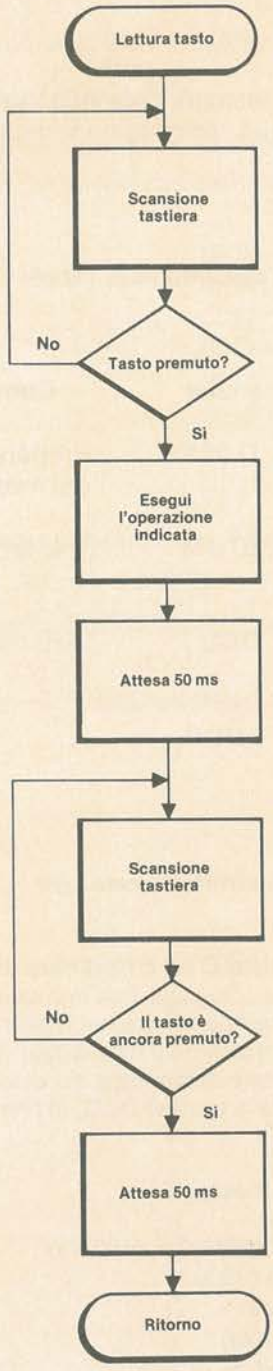


Figura 14-3. Diagramma di flusso del programma antirimbalzo per la tastiera

ESPERIMENTO 14-3

Visualizzazione di un messaggio

INTRODUZIONE

In questo esperimento vengono utilizzate le routine, contenute nella ROM del μLab, al fine di visualizzare un messaggio sul display a LED a sette segmenti. Il messaggio viene preparato caricando in memoria i codici dei caratteri desiderati.

PROCEDIMENTO

A) Codificate e introducete da tastiera il programma descritto nella Tabella 14-6. La routine DCD inizia all'indirizzo 01E9 e quella STDM all'indirizzo 0018.

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	_____		LXI D,0810	;Prepara l'indirizzo
0801	_____			del messaggio
0802	_____			
0803	_____		CALL STDM	;Trasferisci il messaggio
0804	_____			
0805	_____			
0806	_____	LOOP:	CALL DCD	;Visualizza il messaggio
0807	_____			
0808	_____			
0809	_____		JMP LOOP	
080A	_____			
080B	_____			

Tabella 14-6. Programma per visualizzare un messaggio

Questo programma per prima cosa carica nei registri D ed E l'indirizzo di inizio del messaggio. Per fare questo, viene utilizzata l'istruzione LXI D (Load Register Pair Immediate). Questa istruzione carica nel registro E il primo byte posto dopo il codice operativo e nel registro D il secondo byte. Quindi, viene chiamata la routine STDM, in modo da trasferire il messaggio (che inizia all'indirizzo indicato dal contenuto dei registri D ed E) nella locazione di memoria in cui la routine DCD si aspetta di trovarlo. Per ultimo, viene continuamente eseguita la routine DCD, al fine di ottenere il rinfresco del display.

B) Caricate in memoria i seguenti dati che formano il messaggio:

0810	06	(cifra di destra del display)
0811	05	(seconda cifra)
0812	04	(terza cifra)
0813	03	(quarta cifra)
0814	02	(quinta cifra)
0815	01	(cifra di sinistra)

C) Fate partire il programma da 0800. Sul display appare il messaggio 1234 56.

D) Il programma di rinfresco può anche generare un certo numero di caratteri alfabetici. Nella Tabella 14-7 sono mostrati tali caratteri e i loro codici corrispondenti. Premete il tasto di RESET per riportare il controllo al monitor. Cambiate i dati del messaggio nel modo seguente:

0810	10
0811	10
0812	14
0813	12
0814	0E
0815	11

E Fate partire il programma.

F) Utilizzando i caratteri contenuti nella Tabella 14-7 costruitevi ora un vostro messaggio. Premete RESET per riportare il controllo al monitor e memorizzate nelle locazioni 0810-0815 i codici corrispondenti ai caratteri prescelti; quindi fate partire il programma.

Carattere	Codice Hex	Carattere	Codice Hex
0	0	F	F
1	1	(spazio)	10
2	2	H	11
3	3	L	12
4	4	u	13
5	5	P	14
6	6	o	15
7	7	U	16
8	8	-	17
9	9	c	18
A	A	I	19
b	B	B.	1A
C	C	r	1B
d	D	-	1C
E	E		

Tabella 14-7. Codici dei caratteri per la routine DCD

ESPERIMENTO 14-3

(continuazione)

RIASSUNTO

Questo esperimento ha utilizzato due routine del monitor per visualizzare un messaggio sul display. La prima routine (STDM) trasferisce il messaggio nelle locazioni RAM utilizzate dalla routine di scansione display. La seconda routine (DCD) traduce il codice del carattere in codice sette-segmenti e rinfresca il display. Sono stati visualizzati anche alcuni caratteri alfabetici.

ESERCIZIO DI PROGRAMMAZIONE N. 5: USO DELLA TASTIERA E DEL DISPLAY

Scrivete un programma che permetta di visualizzare, nella cifra di sinistra del display, il valore del tasto premuto. Per realizzare tale programma, utilizzate la routine KIND (come nell'Esperimento 14-1) per leggere la tastiera, e le routine STDM e DCD (come nell'Esperimento 14-3) per inviare i dati al display. Ricordate che se il vostro programma memorizza dati in RAM, occorre usare le locazioni 0B00-0B90.

CONTROLLO DIRETTO DEL DISPLAY

Come mostrato in Figura 14-4, il display viene controllato da due porte: una per la scansione delle cifre ed una per la scansione dei segmenti. Ogni bit della porta di scansione controlla una cifra, mentre ogni bit della porta dei segmenti controlla un segmento. Il formato dei dati relativo a queste porte è riportato in Figura 14-4. Per visualizzare un messaggio, occorre in primo luogo convertire i caratteri in codice sette-segmenti. Una dopo l'altra vengono quindi illuminate le cifre, secondo la sequenza definita attraverso la porta di scansione, con i segmenti opportunamente accesi, in modo tale da visualizzare il carattere desiderato.

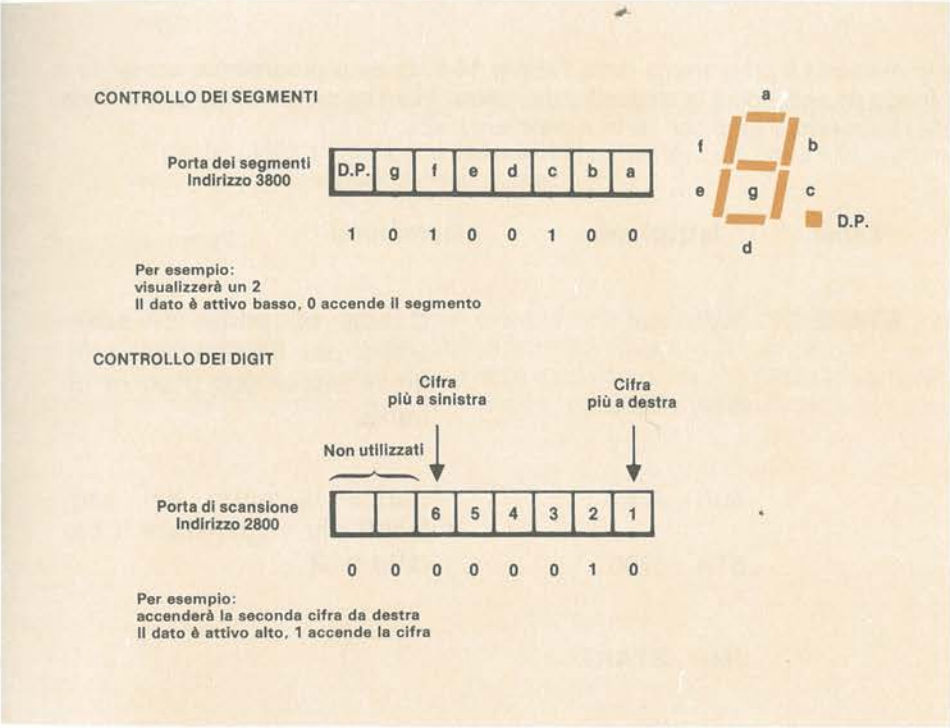


Figura 14-4. Controllo del display del μLab

ESPERIMENTO 14-4

Controllo diretto di un display

INTRODUZIONE

In questo esperimento le porte di scansione e dei segmenti vengono comandate in modo tale da visualizzare un «2» su di una cifra del display. Il dato inviato alle porte è poi modificato: viene così modificato il carattere visualizzato o la sua posizione sul display.

PROCEDIMENTO

A) Codificate e caricate in memoria il programma della Tabella 14-8. Questo programma comanda la porta di scansione in modo da accendere la terza cifra da destra. In un secondo tempo, viene inviato, alla porta che pilota i segmenti, il dato che serve a generare il «2».

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	_____	START:	MVI A,4	;Carica la porta di scansione per selezionare il digit (4 hex = 0000 0100 in binario)
0801	_____			
0802	_____		STA 2800	
0803	_____			
0804	_____			
0805	_____		MVI A,A4	;Carica la porta dei segmenti per visualizzare il carattere "2"
0806	_____			
0807	_____		STA 3800	
0808	_____			
0809	_____			
080A	_____		JMP START	
080B	_____			
080C	_____			

Tabella 14-8. Programma per visualizzare un «2»

B) Fate partire il programma. Nella terza cifra, da destra, viene visualizzato il carattere 2 . Notate che il carattere è più luminoso di quelli normalmente visualizzati. Questo perché, normalmente, vengono spazzolate tutte le sei cifre del display e ciascuna cifra rimane accesa solamente per un sesto dell'intero tempo di scansione, mentre nel nostro caso è una sola cifra che rimane accesa per tutto il tempo.

C) Premete [RESET] per ridare il controllo al monitor. Cambiate in 8 hex (0000 1000 in binario) il dato inviato alla porta di scansione. Questo vi permette di selezionare la quarta cifra da destra.

D) Fate partire il programma. Il carattere si è mosso di una posizione verso sinistra.

E) Fermate il programma e cambiate in 9B il dato inviato alla porta dei segmenti. Cercate di indovinare, con l'aiuto della Figura 14-4, quale sarà il nuovo carattere visualizzato.

F) Fate partire il programma. Viene visualizzato un nuovo carattere. Notate che è possibile generare dei nuovi caratteri a proprio piacere, dal momento che si possono controllare direttamente i singoli segmenti.

G) Premete [RESET]. Scegliete e codificate un nuovo carattere (riferendovi alla Figura 14-4) e cambiate il programma in modo da generarlo effettivamente. Fate partire il programma e controllate che venga visualizzato il carattere che avete scelto.

RIASSUNTO

Senza l'uso delle routine presenti nel monitor, si è controllato direttamente il display a sette-segmenti del μLab. Per selezionare la posizione del carattere è stato inviato un dato alla porta di scansione, mentre, per definire il carattere, è stato inviato un altro dato alla porta che pilota i segmenti. Dato che i segmenti sono controllati l'uno indipendentemente dall'altro, è possibile generare facilmente nuovi caratteri.

SCANSIONE DI TUTTE LE CIFRE

Il passo successivo da compiere, per poter dire che si ha un programma completo di controllo del display, è quello di illuminare tutte le sei cifre. Come detto in precedenza, questo si ottiene facendo accendere le cifre, una dopo l'altra, secondo una sequenza ciclica.

Poiché l'operazione di scrittura delle sei cifre richiede che voi aggiorniate i dati sulle porte dei segmenti e di scansione sei volte durante ogni ciclo di scansione, è molto più comodo, per il programma, disporre di una subroutine che svolga tale compito. La Tabella 14-9 mostra una routine che scrive nella porta di scansione il dato contenuto nel registro B, e il contenuto del registro C nella porta che pilota i segmenti.

Indirizzo	Contenuto	Label	Istruzione	Commenti
0830	3E	DISP:	MVI A,FF	;Spegni tutti i segmenti
0831	FF			
0832	32		STA 3800	
0833	00			
0834	38			
0835	78		MOVA,B	;Carica la porta di scan-
0836	32		STA 2800	sione
0837	00			
0838	28			
0839	79		MOVA,C	;Carica la porta dei seg-
083A	32		STA 3800	menti
083B	00			
083C	38			
083D	C9		RET	

Tabella 14-9. Subroutine display

Per far sì che il dato della cifra precedente non sia visualizzato, per un breve istante occorre per prima cosa spegnere tutti i segmenti. Se non si facesse questo, la porta di scansione farebbe accendere la prossima cifra della sequenza mentre la porta dei segmenti conterrebbe ancora la cifra precedente.

Per accendere tutte e sei le cifre, serve un programma che predisponga i registri B e C e che chiami la routine DISP ogni volta che sia richiesta la visualizzazione di una cifra. Il listing del programma è riportato nella Tabella 14-10. L'istruzione LXI permette ai registri B e C di essere caricati con una sola istruzione, anche se in effetti tali registri vengono utilizzati per scopi diversi. Il primo byte, che segue il codice operativo dell'istruzione LXI, viene caricato nel registro C ed è il dato che deve essere inviato alla porta dei segmenti della routine di display. Il secondo byte, caricato nel registro B, costituisce invece il dato utilizzato per selezionare la cifra da illuminare (inviato alla porta di scansione). La subroutine DISP viene chiamata (Tabella 14-9) dopo che sono stati caricati i registri B e C. Questo programma, senza utilizzare la subroutine del monitor, controlla tutte e sei le cifre del display.

Indirizzo	Contenuto	Label	Istruzione		Commenti
0800	01	START:	LXI	B,018E	;Definisci il dato per la cifra di destra
0801	8E				
0802	01				
0803	CD		CALL	DISP	;Visualizza la cifra
0804	30				
0805	08				
0806	01		LXI	B,0286	;Definisci il dato per la seconda cifra
0807	86				
0808	02				
0809	CD		CALL	DISP	;Visualizza la seconda cifra
080A	30				
080B	08				
080C	01		LXI	B,04A1	;Definisci il dato per la terza cifra
080D	A1				
080E	04				
080F	CD		CALL	DISP	;Visualizza la terza cifra
0810	30				
0811	08				
0812	01		LXI	B,08C6	;Definisci il dato per la quarta cifra
0813	C6				
0814	08				
0815	CD		CALL	DISP	;Visualizza la quarta cifra
0816	30				
0817	08				
0818	01		LXI	B,1083	;Definisci il dato per la quinta cifra
0819	83				
081A	10				
081B	CD		CALL	DISP	;Visualizza la quinta cifra
081C	30				
081D	08				
081E	01		LXI	B,2088	;Definisci il dato per la sesta cifra
081F	88				
0820	20				
0821	CD		CALL	DISP	;Visualizza la sesta cifra
0822	30				
0823	08				
0824	C3		JMP	START	;Ripeti
0825	00				
0826	08				

Tabella 14-10. Programma di scansione display

ESPERIMENTO 14-5

Scansione del display

INTRODUZIONE

Viene caricato ed eseguito il programma precedentemente descritto. Anche se in realtà le cifre si illuminano solo una per volta, sembra che l'intero display sia tutto illuminato nello stesso istante. Questo serve ad illustrare come opera un display in multiplex (scansione delle singole cifre).

PROCEDIMENTO

A) Caricate da tastiera il programma di scansione display riportato nella Tabella 14-10.

B) Caricate la subroutine DISP della Tabella 14-9.

C) Verificate che il programma sia memorizzato correttamente.

D) Fate girare il programma di scansione. Sul display appare AbCd EF.

E) Fermate il programma e cambiate il messaggio. Dovete modificare i dati relativi ai segmenti di ciascuna cifra (memorizzati nelle locazioni 0801, 0807, 080D, 0813, 0819, e 081F). La Figura 14-4 è utile per determinare il codice a sette-segmenti del carattere desiderato.

F) Fate girare ancora il programma in modo da vedere il vostro nuovo messaggio.

RIASSUNTO

È stato caricato ed eseguito un programma che permette di visualizzare un messaggio che copre tutto il display. Questo programma, anche se produce lo stesso risultato del programma di rinfresco del display, presente nel monitor, è però meno sofisticato: vi richiede di specificare il messaggio in codice sette-segmenti e non accetta un messaggio da un qualunque punto della memoria (il messaggio è interno al programma stesso). Nonostante queste differenze, questo programma illustra le soluzioni utilizzate per scandire un display.

La tastiera ed il display del μLab sono scanditi dal software. Nel display viene pilotata una cifra dopo l'altra, in una sequenza ciclica. La tastiera viene pure letta una riga per volta. Il processo di scansione è una soluzione accettabile perché il tempo, impiegato dal microprocessore per scandire tutto il display e la tastiera, è inferiore al tempo di reazione di una persona.

La tastiera e il display possono essere usati molto facilmente utilizzando le routine presenti nel monitor. Risulta più complicato scrivere un programma che realizzi le operazioni di scansione e di conversione codice senza utilizzare le routine del monitor: tale programma vi permette però di vedere in dettaglio queste operazioni e di generare anche dei nuovi caratteri.

La maggior parte dei prodotti basati su microprocessore contengono una tastiera ed un display, che spesso vengono interfacciati utilizzando le tecniche descritte in questa Lezione. Se da un lato è richiesta una certa quantità di software, d'altro canto lo hardware risulta essere molto semplice. Questo è un altro esempio di una funzione realizzata tradizionalmente in hardware, che può essere ora effettuata attraverso dei programmi software.

DOMANDE

Lezione 14

1. Se vengono premuti i tasti «2» e «3», il programma della Tabella 14-5 genererà:
   a. un suono di avvertimento finché il tasto «2» rimane premuto.
   b. un suono di avvertimento finché il tasto «3» rimane premuto.
   c. nessun suono finché rimane premuto il tasto «3».
   d. un suono di avvertimento finché uno dei due tasti rimane premuto.

2. Se col programma antirimbalzo di Figura 14-3 viene usato un tasto che rimbalza per 15 ms, il sistema:
   a. eliminerà correttamente i rimbalzi del tasto.
   b. potrebbe credere che il tasto sia stato premuto due volte anziché una.
   c. potrebbe credere che il tasto sia stato premuto tre o più volte.
   d. potrebbe credere che è stato premuto un tasto diverso.

3. Il display è illuminato:
   a. un segmento per volta.
   b. una cifra per volta.
   c. due cifre per volta.
   d. tutto in una volta sola.

4. Una funzione, che è stata realizzata dalla routine di scansione display DCD presente nel monitor, ma non dal programma della Tabella 14-10, è:
   a. la scansione di tutte e sei le cifre.
   b. il controllo di tutti e sette i segmenti.
   c. la conversione in codice sette-segmenti.
   d. l'eliminazione dei rimbalzi dei tasti.

LEZIONE 15

Rappresentazioni numeriche e algoritmi

INTRODUZIONE

Come i calcolatori elettronici, anche i sistemi basati sui microprocessori vengono utilizzati frequentemente per realizzare delle elaborate funzioni matematiche su diversi tipi di numeri. Tuttavia, nei programmi che sono stati finora utilizzati in questo corso, ci si è serviti solo dei numeri interi positivi compresi tra 0 e 255, e, tra le funzioni, si sono viste solo quelle logiche, la addizione e la sottrazione. In questa lezione vengono discusse le tecniche utilizzate per rappresentare un campo numerico più esteso e per realizzare operazioni matematiche complesse.

NUMERI NEGATIVI

Utilizzando una parola di otto bit, cerchiamo di vedere come sia possibile rappresentare numeri interi sia positivi che negativi. Dal momento che, utilizzando otto bit, non è possibile rappresentare più di 256 numeri diversi il campo numerico verrà limitato a ± 127. Come positivi sono definiti i primi 128 numeri, da zero a 127 (7F hex). I numeri negativi sono generati partendo da zero e contando «all'indietro». Come nel caso di un contatore hardware di tipo up/down (su/giù), se un registro contiene 0000 0000, quando viene decrementato di uno il suo valore diventa 1111 1111 (FF hex). Perciò —1 è uguale ad FF hex. Tale rappresentazione prende il nome di *complemento a due*. La rappresentazione in complemento a due dei numeri compresi tra —8 e +7 è data in Tabella 15-1.

Decimali	Complemento a due
7	0000 0111
6	0000 0110
5	0000 0101
4	0000 0100
3	0000 0011
2	0000 0010
1	0000 0001
0	0000 0000
—1	1111 1111
—2	1111 1110
—3	1111 1101
—4	1111 1100
—5	1111 1011
—6	1111 1010
—7	1111 1001
—8	1111 1000

Tabella 15-1. Rappresentazione in complemento a due dei numeri da —8 a +7

Osservate che il bit più significativo (MSB) indica il segno del numero. Se MSB = 0 il numero è positivo, se MSB = 1 il numero è negativo.

Il procedimento da utilizzare per ricavare la rappresentazione complemento a due è particolarmente semplice. Per i numeri positivi, la rappresentazione in complemento a due coincide con quella binaria (come è evidente dai primi otto numeri della Tabella 15-1). Per i numeri negativi, diamo di seguito il procedimento per calcolare la rappresentazione in complemento a due:

1. Scrivete il numero, espresso in valore assoluto (p. es., per —5 scrivete 0000 0101), in codice binario.

2. Complementate il numero binario (ottenete così il *complemento ad uno*, es. $\overline{0000\ 0101}$ = 1111 1010).

3. Per formare il complemento a due, sommate infine uno (p. es., 1111 1010 + + 1 = 1111 1011 = —5 in complemento a due).

Analogamente se è dato un numero negativo, espresso in complemento a due, e si vuole ottenere il suo valore assoluto occorre complementare il numero e sommare uno.

Per esempio, considerato il numero in complemento a due 1111 1011:

$$\overline{1111\ 1011} = 0000\ 0100 \quad 0000\ 0100 + 1 = 0000\ 0101 = 5$$

Perciò, 1111 1011 è la rappresentazione in complemento a due di meno cinque. Notate che se avessimo considerato il numero 1111 1011 come binario puro e non in complemento a due esso sarebbe stato interpretato come il numero decimale 251. È perciò necessario segnalare che un dato è espresso in complemento a due e ricordarsi di utilizzarlo di conseguenza. Per le operazioni aritmetiche, la rappresentazione in complemento a due è molto comoda.

I numeri in complemento a due, quando sono divisi, moltiplicati, sommati o sottratti, danno ancora per risultato dei numeri in complemento a due. Questa rappresentazione è comunemente usata in sistemi a microprocessore che devono utilizzare dei numeri sia positivi che negativi.

PICCOLI E GRANDI NUMERI

Anche se permette di rappresentare i numeri negativi, la rotazione complemento a due lascia però il campo degli interi rappresentabili limitato a numeri di valore assoluto minore di 129. A seconda della dimensione del campo richiesto e del grado di precisione necessaria, si può estendere tale campo in molti modi diversi.

Doppia precisione

Il modo più semplice per estendere il campo dei numeri rappresentabili è quello di aumentare il numero di bit usati per rappresentare ciascun numero. Questo viene spesso fatto utilizzando una coppia di parole per rappresentare un unico numero (Figura 15-1). Con il microprocessore 8085, tale soluzone è facilitata dalla presenza di istruzioni dedicate alle coppie di registri, che operano contemporaneamente su sedici bit. L'uso di due parole per rappresentare un numero è detto *doppia precisione* (Double Precision). Con un processore a otto bit, la tecnica della doppia precisione estende il campo da 0 a 65.535 ovvero ± 32.767.

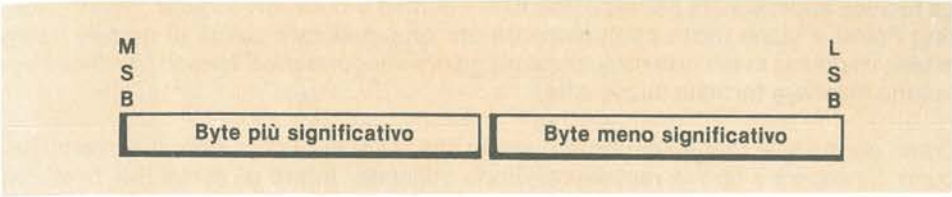


Figura 15-1. Doppia precisione

Virgola fissa

Se è vero che la doppia precisione aumenta il campo dei numeri rappresentabili, che cosa succede ai numeri minori di uno, o compresi tra 3 e 4? La Figura 15-2 mostra un'ulteriore rappresentazione, detta a *virgola fissa* (Fixed Point). In questo esempio, per memorizzare un numero vengono utilizzati due byte. Il primo byte corrisponde alla parte intera del numero, quella a sinistra della virgola (è in realtà una virgola binaria), mentre il secondo byte corrisponde alla parte frazionaria (la parte a destra della virgola). In questo modo è possibile rappresentare numeri piccoli fino a $2^{-8} = 1/256$, o numeri frazionari come 3,17. Tuttavia la risoluzione è limitata a 1/256 (circa 0,004) ed i valori massimi sono limitati tra ± 127.

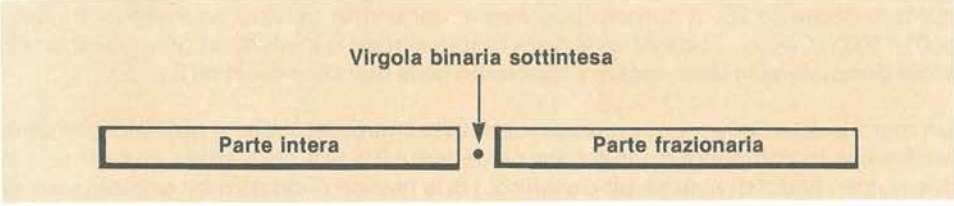


Figura 15-2. Virgola fissa

Virgola mobile

È possibile estendere la capacità di rappresentazione della virgola fissa utilizzando più byte per ciascuna delle due parti del numero. Se però non si vuole dedicare a ciascun numero una notevole quantità di memoria, resta ancora impossibile rappresentare dei numeri come 360.000.000.000 o 0,000000297. In questi tipi di numeri sono presenti molti zeri che identificano solo la grandezza del numero. Utilizzando la «notazione scientifica», in cui son utilizzati una *mantissa* ed un *esponente*, è però possibile rappresentare questi numeri in modo assai semplice. La mantissa è un valore che esprime quanto il numero è grande, modificato in modo da essere compeso tra 0 ed 1. Per esempio 360.000.000.000 può essere scritto come $0{,}36 \times 10^{12}$ (in questo caso 0,36 è la mantissa, compresa tra 0 e 1, e 12 è l'esponente); mentre 0,000000297 si può scrivere come $0{,}297 \times 10^{-6}$ (in questo caso la mantissa è 0,297 e -6 l'esponente).

Supponiamo ora che per rappresentare un numero si utilizzino due byte, uno per la mantissa e uno per l'esponente, come è illustrato in Figura 15-3. Supponendo che sia la mantissa che l'esponente vengano memorizzati come numeri in complemento a due, il campo di valori che possono essere rappresentati sarà circa $\pm 10^{127}$. Il campo così ottenuto è molto esteso; infatti 10^{127} è un numero molto grande mentre 10^{-127} è molto piccolo.

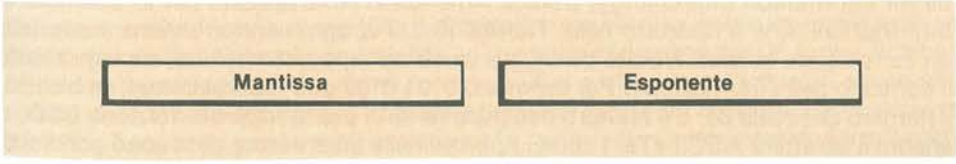


Figura 15-3. Virgola mobile

La tecnica appena vista per rappresentare i numeri è chiamata *virgola mobile* (Floating Point) e viene usata comunemente per rappresentare campi di numeri molto estesi. Per poter avere una risoluzione maggiore vengono usati spesso più di due byte (una mantissa formata da più cifre).

Come per tutte le rappresentazioni, notate che, se si vuole decifrare il numero, occorre conoscere il tipo di rappresentazione utilizzata. Infatti gli stessi due byte che esprimono il dato possono essere interpretati ben diversamente a seconda che vengano letti come coppia di numeri espressi in complemento a due, o come un unico numero in virgola fissa o come un unico numero in virgola mobile. Il programma che elabora i dati deve quindi conoscere quale tipo di rappresentazione è utilizzata.

RAPPRESENTAZIONE DEI NUMERI DECIMALI

La maggior parte dei sistemi a microprocessore utilizza dispositivi di I/O, ad esempio tastiera e video, di tipo decimale. (Il μLab costituisce una eccezione, dal momento che usa una notazione esadecimale). Essendo la rappresentazione decimale quella più naturale per le persone, molti sistemi a microprocessore devono tenerne conto.

Il problema è come sia possibile rappresentare numeri decimali in un sistema che opera con dei numeri binari. Supponiamo, per esempio, che venga letto da tastiera il numero decimale 28. Il numero può essere convertito nel suo equivalente binario, 0001 1100 (1C hex). Tuttavia se si deve rappresentare quest'ultimo numero su un display decimale esso deve essere riconvertito nelle due cifre decimali 2 ed 8.

Un metodo alternativo, per rappresentare un numero decimale, è quello di prendere le cifre che lo compongono (nel caso precedente 2 e 8) e convertirle una per una in due numeri binari di quattro bit ciascuno. I due numeri di quattro bit possono quindi essere compattati in unico byte. Così 28 verrebbe codificato in 0010 1000. Questa tecnica viene chiamata *Binary Coded Decimal* (BCD, Decimale Codificato in Binario). Notate che i valori binari compresi tra 1010 e 1111 non vengono mai usati nella rappresentazione BCD.

La rappresentazione BCD è comunemente adottata per sistemi che utilizzano I/O di tipo decimale poiché permette di eliminare l'operazione di conversione decimale-binario. È però svantaggiosa per quanto riguarda la quantità di memoria occupata; infatti, il più grande numero decimale rappresentabile in BCD con un byte è 99, mentre in binario puro si avrebbe 255. Siccome la rappresentazione BCD non è «naturale», ne risulta che l'aritmetica corrispondente diviene piuttosto complessa. La maggior parte dei microprocessori tuttavia è dotata di istruzioni particolari in grado di operare su numeri BCD. (Osservate nella Appendice B la descrizione dell'istruzione DAA).

RAPPRESENTAZIONE DEI CARATTERI ALFANUMERICI

La maggior parte dei sistemi a microprocessore deve essere in grado di trattare anche i simboli alfabetici e non solo i numeri. Per esempio, un terminale collegato ad un calcolatore deve poter leggere i caratteri provenienti dalla tastiera ed inviarli al calcolatore. Anche le lettere devono perciò essere in qualche modo rappresentate tramite numeri binari.

Il codice più comune, a tal fine utilizzato, è chiamato ASCII (American Standard Code for Information Interchange, Codice Americano Normalizzato per lo Scambio di Informazioni) che è riportato nella Tabella 15-2. Ad ogni carattere viene assegnato un certo valore binario. Notate come, per qualsiasi rappresentazione, sia importante il contesto dell'informazione. Per esempio, 0101 0100 può rappresentare, in binario, il numero decimale 84, o il numero decimale 54 se si usa la rappresentazione BCD, o ancora il carattere ASCII «T». I codici riportati nelle aree tratteggiate sono «codici di controllo» cui corrispondono funzioni particolari. Per esempio, il codice «0A» è usato per effettuare una interlinea su una stampante o su un video.

00	NUL	20	SPACE	40	@	60	‘
01	SOH	21	!	41	A	61	a
02	STX	22	”	42	B	62	b
03	ETX	23	#	43	C	63	c
04	EOT	24	$	44	D	64	d
05	ENQ	25	%	45	E	65	e
06	ACK	26	&	46	F	66	f
07	BEL	27	’	47	G	67	g
08	BS	28	(	48	H	68	h
09	HT	29	)	49	I	69	i
0A	LF	2A	*	4A	J	6A	j
0B	VT	2B	+	4B	K	6B	k
0C	FF	2C	,	4C	L	6c	l
0D	CR	2D	—	4D	M	6D	m
0E	SO	2E	.	4E	N	6E	n
0F	SI	2F	/	4F	O	6F	o
10	DLE	30	0	50	P	70	p
11	DC1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	72	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	SUB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D	}
1E	RS	3E	>	5E	^	7E	~
1F	US	3F	?	5F	—	7F	DELETE

I CODICI NELLE AREE COLORATE SONO CODICI DI CONTROLLO

Tabella 15-2. Codici ASCII

Essendo arbitraria la corrispondenza tra i caratteri e i codici, esistono molte altre rappresentazioni possibili. Il codice ASCII è attualmente quello più utilizzato, ma nel passato, per esempio, veniva frequentemente utilizzato un altro codice chiamato *BAUDOT*. Le macchine IBM usano il codice *EBCDIC* (Extended Binary Coded Decimal Interchange Code, Codice Decimale di Scambio Codificato in Binario Esteso).

TABLE LOOK-UP

Un tipico problema di programmazione riguarda la conversione di un numero, espresso in una certa rappresentazione (codice), nel suo equivalente in un'altra rappresentazione. Supponete, per esempio, di dover visualizzare una cifra esadecimale su un display a sette/segmenti. In un qualche modo occorre determinare quali sono i segmenti che devono essere accesi, in modo da poter visualizzare il carattere prescelto. In questo caso occorre convertire il codice binario in quello sette-segmenti.

Questa conversione viene eseguita utilizzando una tecnica chiamata *table look-up* (lett.: dare un'occhiata ad una tabella). In memoria sono elencate, in forma di tabella, le configurazioni dei segmenti corrispondenti a ciascun carattere. Il primo elemento della tabella contiene la configurazione dei segmenti corrispondenti al carattere «0», il secondo quella per il carattere «1», e così via. Onde poter tradurre il codice binario nel corrispondente codice sette-segmenti, è sufficiente «dare un'occhiata» alla tabella.

Nella Figura 15-4 è presentato il diagramma di flusso di un programma che converte i dati binari in codice sette-segmenti. Questo programma si serve di una tabella di codici sette-segmenti. La prima locazione della tabella contiene il codice sette-segmenti corrispondente a «0»; la seconda contiene il codice sette-segmenti di «1». Per prima cosa, il numero binario da convertire viene sommato all'indirizzo della prima locazione della tabella. Il risultato dà l'indirizzo del punto della tabella in cui è contenuto il codice sette-segmenti desiderato. Per completare l'operazione di conversione, basta leggere il contenuto della locazione indirizzata.

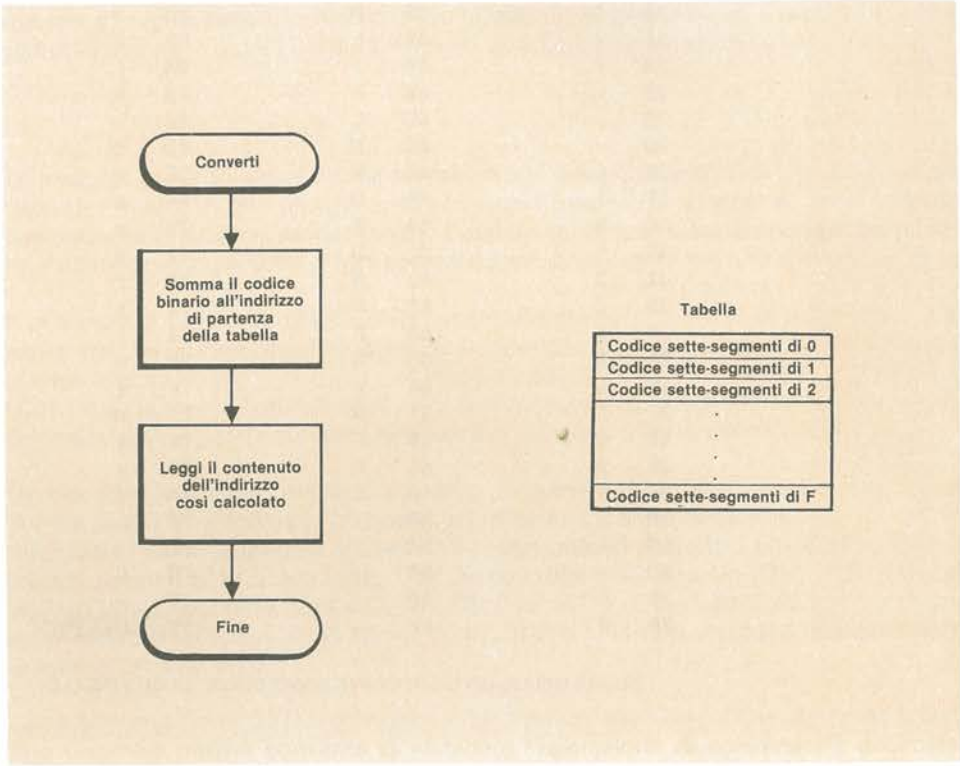


Figura 15-4. Programma con accesso in tabella per la conversione da binario a sette-segmenti

ALGORITMI MATEMATICI

A questo punto potreste chiedervi come è possibile realizzare delle funzioni complesse utilizzando solamente l'insieme disponibile di istruzioni. Le calcolatrici, che contengono dei microprocessori, possono effettuare divisioni, moltiplicazioni, seni, coseni e molte altre funzioni matematiche. Questo è possibile anche se, direttamente, il microprocessore 8085 (e così molti altri processori a 8 bit) può eseguire, come operazioni aritmetiche, solo l'addizione e la sottrazione.

In effetti, le funzioni matematiche complesse sono realizzate (o approssimate) utilizzando solamente le istruzioni di somma e sottrazione. La moltiplicazione può essere semplicemente realizzata eseguendo una serie di somme e di shift. Per vedere come questo sia possibile, considerate il modo in cui voi, «manualmente», eseguite la moltiplicazione. Il moltiplicando viene moltiplicato da una cifra del moltiplicatore per volta. I diversi risultati che si ottengono per ogni cifra del moltiplicatore sono spostati progressivamente a sinistra di un posto e vengono infine sommati. La stessa tecnica può essere usata per i numeri binari. In questo caso la moltiplicazione per una sola cifra è banale: un numero moltiplicato per uno rimane inalterato, ed un numero moltiplicato per zero vale zero (Figura 15-5). Quindi l'intera moltiplicazione può essere eseguita usando operazioni di shift e di somma. Con un processo analogo, che fa uso delle operazioni di shift e di sottrazione, viene realizzata la divisione. Le tecniche che portano a svolgere una data operazione sono chiamate *algoritmi*.

```
        1011
x       1001
------------
        1011
       0000
      0000
     1011
------------
     1100011
```

Figura 15-5. Moltiplicazione binaria

Il calcolo di una funzione come il seno di un angolo è un'operazione più complicata, ma esistono algoritmi che approssimano in modo molto soddisfacente la funzione seno, utilizzando solamente delle funzioni semplici, per esempio:

$$\text{seno } X = X - \frac{X^3}{3\times 2} + \frac{X^5}{5\times 4\times 3\times 2} - \frac{X^7}{7\times 6\times 5\times 4\times 3\times 2} + \frac{X^9}{9\times 8\times 7\times 6\times 5\times 4\times 3\times 2} - \ldots$$

Questa è una serie infinita, che darà il valore esatto del seno di un numero solo se viene preso un numero infinito di termini. Contiene moltiplicazioni e divisioni, ma operazioni, abbiamo già visto, possono essere realizzate utilizzando delle istruzioni di somma e di sottrazione.

In pratica, potrà naturalmente essere calcolato solo un numerò finito di termini. Questo comporta che il risultato avrà un valore approssimato, ma se vengono calcolati molti termini, cosa che non costituisce un problema per un microprocessore, il risultato ottenuto potrà essere assai preciso.

La serie sopra riportata ci dà un algoritmo per il calcolo del seno di un angolo. Per ogni funzione matematica esiste un algoritmo che permette di calcolare la funzione stessa, utilizzando solamente quelle operazioni elementari che possono essere eseguite da un microprocessore.

RIPASSO

Lezione 15

Usando il sistema standard di numerazione binaria i processori a 8 bit sono limitati agli interi compresi tra 0 e 256. I numeri negativi possono essere rappresentati utilizzando la rappresentazione in complemento a due, che assegna metà dei valori rappresentabili ai numeri positivi e metà ai numeri negativi.

Numeri più grandi possono essere rappresentati in vari modi. Il più semplice è la doppia precisione che utilizza due byte per rappresentare ciascuna parola dei dati. In questo modo si hanno a disposizione sedici bit, ed il campo dei numeri rappresentabili va così da zero a 65.535. Per campi ancora più estesi si utilizza la rappresentazione in virgola mobile. Un byte è utilizzato per rappresentare la mantissa e un altro per l'esponente. Nella rappresentazione in virgola fissa, infine, si utilizza un byte per la parte intera ed uno per la parte frazionaria.

Per rappresentare dei numeri in una forma che sia conveniente per ingressi ed uscite decimali, si ricorre ai codici BCD. Ciascuna cifra decimale è convertita, in modo separato, in un numero binario di quattro bit.

Per rappresentare dei caratteri alfanumerici viene assegnato un codice binario a ciascun carattere. Il codice ASCII è attualmente il più utilizzato.
Utilizzando algoritmi, si possono realizzare delle funzioni matematiche complesse. Una qualsiasi funzione può essere approssimata (se non calcolata esattamente) tramite una serie di somme e sottrazioni.

1. 1111 0001 è la rappresentazione in complemento a due di:
   a. F1 (hex).
   b. —F1.
   c. 0E.
   d. —0F.

2. Qual è il più grande numero rappresentabile in doppia precisione, per un processore che utilizza parole di quattro bit:
   a. 15 (decimale).
   b. 255.
   c. 127.
   d. 65.535.

3. Si possono rappresentare numeri molto grandi e numeri molto piccoli utilizzando la rappresentazione _____.

4. La rappresentazione BCD viene utilizzata perché:
   a. semplifica l'I/O decimale.
   b. semplifica i calcoli.
   c. la memoria viene usata in modo più efficiente.
   d. si possono realizzare funzioni artimetiche complesse.

5. Le funzioni aritmetiche complesse sono realizzate dal microprocessore:
   a. utilizzando istruzioni speciali progettate appositamente per la funzione.
   b. utilizzando algoritmi basati su somme e sottrazioni.
   c. utilizzando solamente particolari chip calcolatori.
   d. solamente se il processore è progettato in modo particolare.

V
RICERCA DEI GUASTI NEI SISTEMI A MICROPROCESSORE

Nel 1979 è stato valutato che in tutto il mondo, nei prodotti di tipo non «consumer» (prodotti di largo impiego), sono contenuti miliardi di circuiti integrati digitali. Se la percentuale media di guasto fosse dello 0,1% all'anno, si dovrebbero rompere 10 milioni di circuiti integrati all'anno. Se poi, valutiamo il costo medio di riparazione in 100 $ per guasto, annualmente verrebbe speso 1 miliardo di dollari solo per riparazioni effettuate presso l'utente. Tutto questo comporta per le sole riparazioni un costo per integrato di 10 centesimi di dollaro. Il costo della manutenzione, ripartito sull'intera vita del prodotto, può ben presto superare lo stesso costo medio degli integrati. Di conseguenza collaudi e tecniche di riparazione più efficienti possono permettere un sostanziale risparmio sui costi.

Con i microprocessori si possono realizzare prodotti con affidabilità, prestazioni, caratteristiche e sofisticazioni decisamente migliori. Ma accanto a questi miglioramenti sono anche sorti dei nuovi problemi connessi con la ricerca guasti, la riparazione e la manutenzione. Per poter affrontare i guasti che si possono verificare in questi complessi sistemi a microprocessore, sono necessari perciò nuovi strumenti e nuove tecniche.

Nelle lezioni 16 e 17 vengono discussi gli strumenti utilizzati per affrontare il problema dei guasti nei circuiti basati sui microprocessori: sonde logiche, generatori di impulsi logici, analizzatori di stato, analizzatori di firma, rivelatori di corrente e oscilloscopi. Nella Lezione 18 verranno poi descritte le teorie per la ricerca dei guasti e le procedure per la loro riparazione. Nella Lezione 19 infine vengono utilizzate queste tecniche e questi strumenti applicandoli al problema della manutenzione del μLab.

La Lezione 16 può anche essere letta velocemente dallo studente che sia già familiarizzato con le sonde logiche, i generatori di impulsi, e i rivelatori di corrente. Ancora, la Lezione 17 può essere letta velocemente solo se si possiede una precedente esperienza con analizzatori logici e con analizzatori di firma.

LEZIONE 16

Strumenti portatili per la ricerca dei guasti

INTRODUZIONE

Le *sonde logiche*, i *generatori di impulsi logici*, e i *rivelatori di corrente* sono strumenti portatili progettati per stimolare e rilevare il comportamento digitale dei circuiti logici. Sono facili da usare, anche se la loro struttura interna è complessa, e questi tre strumenti si dimostrano assai efficaci in un largo spettro di situazioni in cui sia necessario riparare dei sistemi digitali. Negli esperimenti verrà illustrato come sia possibile utilizzare tali strumenti, sia singolarmente che in combinazione, per semplificare la localizzazione dei guasti in un circuito digitale controllato con microprocessore.

ESPERIMENTO 16-1

Sonde logiche

INTRODUZIONE

Le sonde logiche (Logic Probe) analizzano il comportamento logico del circuito. Per mezzo di una semplice lampadina di indicazione, vi mostrano lo stato logico e vi permettono di rilevare anche brevi impulsi. In questo esperimento, verranno usati i segnali logici del μLab per illustrare come la sonda logica HP 545A opera e come essa possa essere positivamente usata per ricercare i guasti.

Sonda logica HP 545A

PROCEDIMENTO

I. Preparazione della sonda logica

A) Assicuratevi che il μLab sia acceso e che sul visualizzatore appaia uLAb UP. Se questo non si verifica, controllate l'interruttore di accensione e premete [RESET].

B) Collegate ad un alimentatore, che generi la stessa tensione del circuito logico sotto esame, i due cavetti di alimentazione della sonda. L'alimentatore del dispositivo sotto esame può essere utilizzato se è in grado di fornire la corrente addizionale (70 mA) richiesta dalla sonda. Ci si può collegare all'alimentazione o utilizzando le pinzette a molla fornite con la sonda o collegandosi direttamente ai piedini di una test clip (pinzetta multipiedini per collaudo) per IC posta su un circuito integrato dello strumento sotto esame (si veda Figura 16-1).

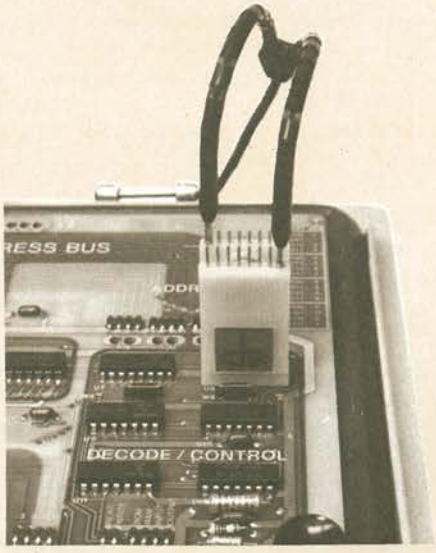


Figura 16-1. Collegamento dell'alimentazione della sonda effettuata attraverso uno zoccolo per il test dei circuiti integrati (IC test clip)

C) Nella Figura 16-2 è illustrato come gli «uncini» metallici delle pinzette a molla possano essere inseriti nei fori di alimentazione (uno grande e uno piccolo) presenti sul μLab. Questi fori si trovano lungo il bordo superiore della scheda del Microprocessor Lab. Collegate il filo rosso di alimentazione della sonda ad una delle coppie di fori di destra (+5) e il filo nero ad una delle coppie di fori di sinistra (↓). Notate che la lampada sul puntale della sonda è poco illuminata, indicando così un cattivo livello di segnale o un circuito aperto (floating, fluttuante).

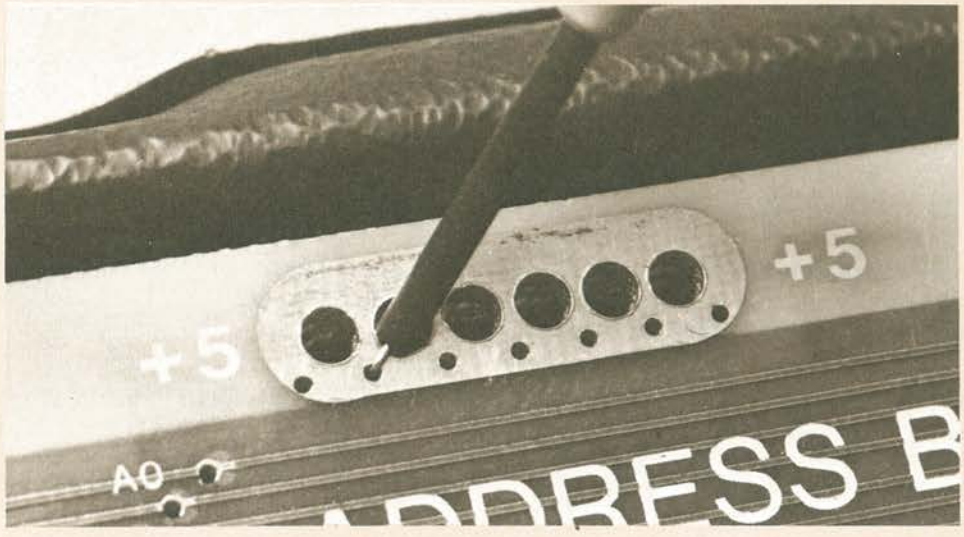


Figura 16-2. Come si usano gli «uncini» per collegare la sonda ai fori di alimentazione del μLab

D) L'interruttore a slitta, posto sulla sonda, serve a predisporre il valore della soglia logica di ingresso ad un livello CMOS o a un livello TTL. Nella posizione CMOS, questi valori variano in funzione della tensione di alimentazione. Nella posizione TTL le soglie invece sono fisse. Quando cercate dei guasti sul μLab, e sulla maggior parte dei sistemi a microprocessore che sono alimentati a 5 Volt, assicuratevi che l'interruttore sia nella posizione TTL. Anche se utilizzano circuiti MOS, tali sistemi vengono progettati per avere, ai piedini, tensioni di soglia di tipo TTL.

II. Uso della sonda

A) La lampadina posta sulla punta della sonda fornisce informazioni sugli stati logici. Osservate che, quando il puntale metallico della sonda non è a contatto con nulla, la lampadina è solo leggermente accesa, indicando una condizione di livello cattivo o di nodo fluttuante. Ogni punto del circuito è chiamato *nodo*. Tutti i punti che sono direttamente collegati fanno parte di uno stesso nodo. Se, quando la sonda tocca un nodo la lampadina è accesa solo leggermente, significa che è rilevato un livello logico non valido (Figura 16-3; il livello si trova cioè tra le soglie degli stati logici «0» e «1»). Un livello floating potrebbe essere una condizione accettabile se il nodo fosse una linea di un bus a tre stati, in cui tutti i dispositivi di uscita dal bus fossero disabilitati (aperti), o se si trattasse di un ingresso aperto di una porta. Si ha invece una condizione anomala se il livello fluttuante viene indicato su di un nodo che dovrebbe corrispondere ad una uscita logica.

Figura 16-3. Soglie di tensione TTL per la sonda logica 545A

B) Controllate le sonde toccando con la punta il contatto di massa. La luce sul puntale si spegne ad indicare un livello logico 0. Ora toccate con la punta il contatto di +5V. La lampadina si illumina, indicando in questo caso un livello logico 1.

C) Usate la sonda per verificare la continuità della linea D7 (posta appena sotto il display all'estrema destra, Figura 16-4), che collega il piedino IC13-18 della porta d'ingresso con il relativo interruttore della porta d'ingresso. Ponete la punta della sonda su IC13-18 e muovete verso l'alto e poi verso il basso l'interruttore d'ingresso posto all'estrema sinistra (ingresso 7). Osservate come la luce segua il cambiamento del livello logico.

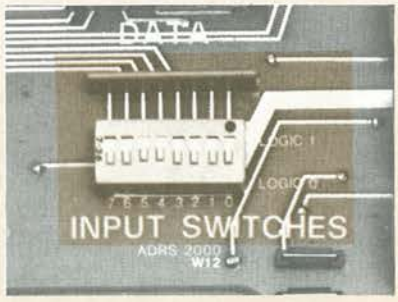


Figura 16-4. Interruttori della porta d'ingresso del μLab

D) Toccate IC18-12 (riferitevi allo schema di Figura 16-5). Questa è una linea di ingresso della tastiera. Osservate che la luce è illuminata in modo costante. Questo vi dice che il segnale si trova stabilmente nello stato logico 1. Questo è vero, perché nessuno dei tasti 0-D (0,1,4,7,A,D) di tale colonna è stato premuto.

E) Provate ora con la sonda tutte le linee di uscita della porta di scansione IC17 (Figura 16-5). La luce lampeggiante indica la presenza di una notevole attività logica su tutte queste linee. Per permet-

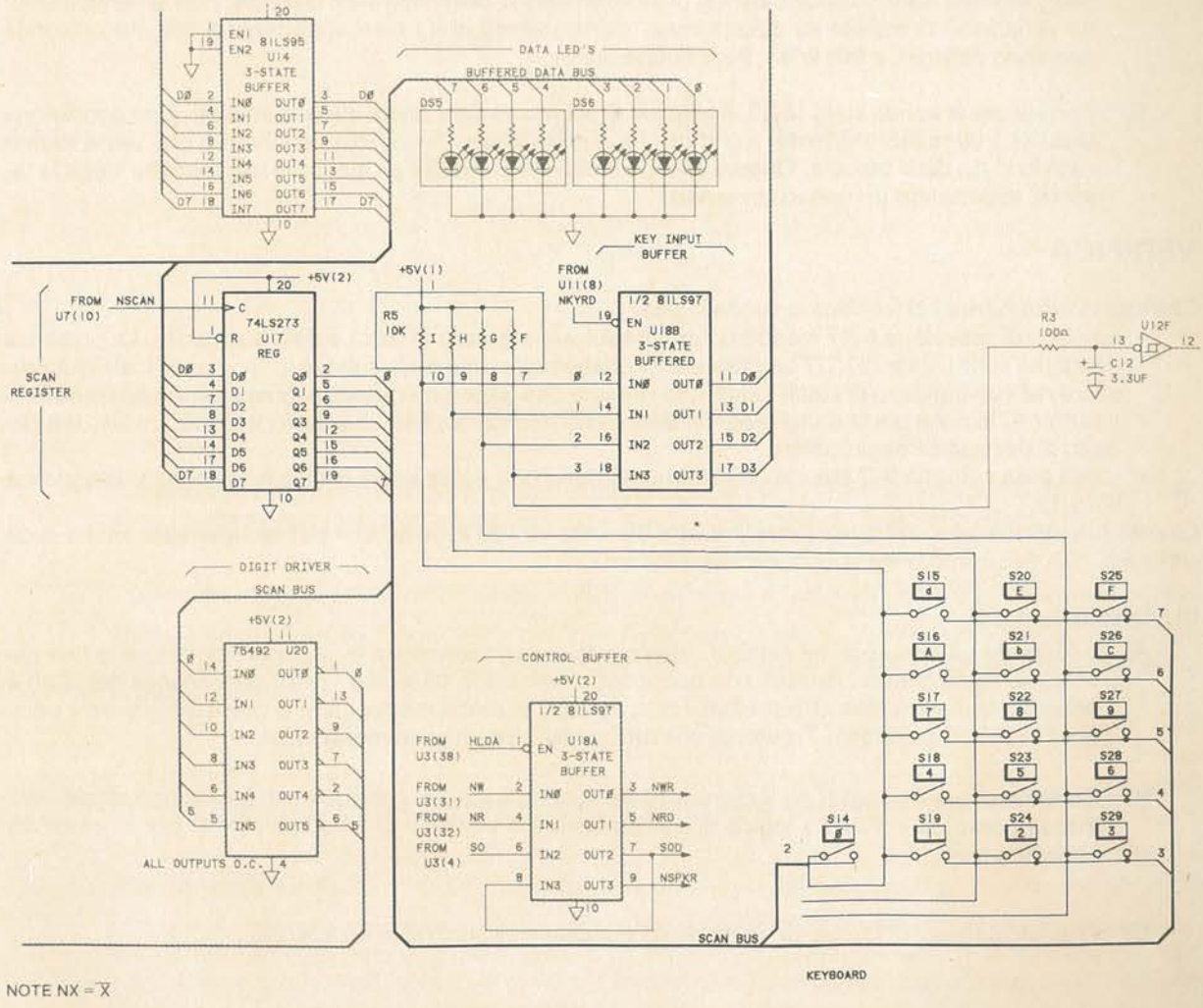


Figura 16-5. Circuito di scansione e di lettura tasti relativo alla colonna dei tasti 0-D

tervi di osservare quanto accade, la sonda logica rallenta la frequenza di lampeggio. In questo modo, se una delle linee fosse inattiva, tale condizione verrebbe chiaramente evidenziata dalla sonda logica.

F) Ora riportatevi nuovamente su IC18-12. Mentre tenete la sonda su questo piedino, premete un qualsiasi tasto della colonna 0-D e osservate che la luce diventa lampeggiante. Questo vi indica che il segnale è stato modificato dai tasti. Premendo i tasti delle altre colonne, non si verifica alcuna variazione di attività su questa linea, dal momento che i tasti appartenenti alle altre colonne non sono collegati a tale linea. Fate alcune prove.

G) Portate ora la sonda su IC18-19, il segnale di selezione della porta d'ingresso tasti. Una condizione di attività (luce intermittente) significa che questo dispositivo è stato abilitato e che viene esaminato lo stato della tastiera. Questo è vero perché sta girando il programma monitor che legge la tastiera, aspettando un nuovo comando.

VERIFICA

Che tipo di informazioni ci fornisce la sonda logica?

- La porta di scansione IC17 mostra un'attività sul suo ingresso di clock e sulle sue uscite. La presenza di attività sulle uscite di IC17 segnala che probabilmente tale dispositivo è in funzione. L'attività presente nel suo ingresso di clock indica che i circuiti che generano questo segnale stanno operando.
- Il buffer IC18 della porta d'ingresso dei tasti è abilitato dal segnale di selezione porta, inviato dal circuito di decodifica degli indirizzi.
- I tasti della colonna 0-D stanno correttamente operando come interruttori e forniscono un segnale a IC18-12.

Quanto possano essere utili queste informazioni dipende sia dall'attività, che potete osservare anche sugli altri nodi, che da quanto bene voi conosciate il sistema.

III. Analisi dei nodi

A) Analizzate gli altri nodi del circuito, utilizzando come riferimento lo schema riportato alla fine del volume. Confrontate i risultati con quanto vi aspettate in base alla vostra conoscenza del μLab e del programma monitor. Controllate i bus, i piedini di selezione dei chip, le porte d'ingresso e uscita e i circuiti di controllo. Troverete che tutti questi circuiti sono molto attivi.

B) Controllate ora il punto IC10-9 (Figura 16-6). Questo è il bit di controllo per la funzione single-step (passo passo). Se il livello logico di questa linea è 0 significa che il μLab non si trova nel modo hardware step.

C) Premete il tasto [FETCH PC] sul μLab, in modo da prelevare e visualizzare un indirizzo.

D) Premete [HDWR STEP] e osservate che il nodo IC10-9 si porta nello stato logico 1. Questo livello indica che il μLab si trova ora nel modo hardware step.

E) Controllate ora il nodo IC10-5. Un livello basso su questa linea, collegata all'ingresso Ready (IC3-35), ferma l'attività sul bus.

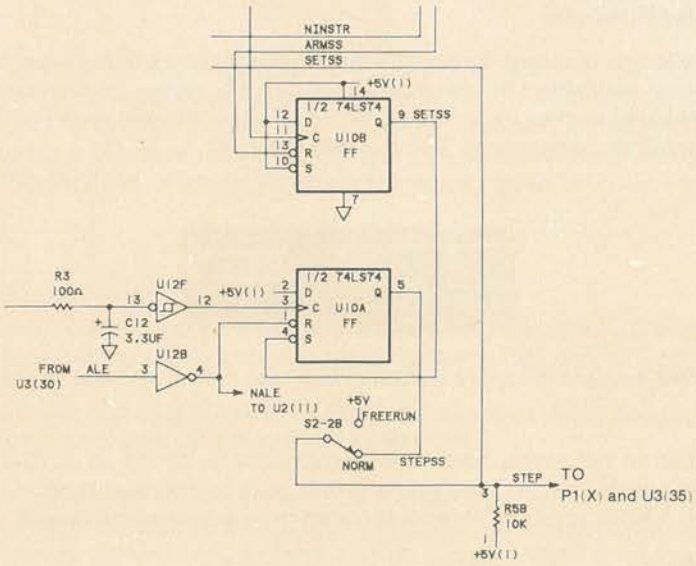


Figura 16-6. Circuito di controllo del single-step.

F) Riesaminate alcuni dei nodi già provati e verificate che la maggior parte dell'attività del μLab è cessata. L'unico nodo logico attivo del sistema è quello.del segnale di clock sul piedino 37 di IC3.

G) Dopo aver posto la punta della sonda su IC10-5, premete il tasto [HDWR STEP] e osservate come la lampada della sonda lampeggi ogni volta che viene premuto un tasto. Sebbene questo impulso abbia una durata di pochi microsecondi, esso appare molto più lungo perché la sonda allunga l'impulso in modo osservabile.

H) Se premete il tasto [HDWR STEP] e osservate i piedini 1,3 e 5 di IC10, potete seguire i segnali logici utilizzati per realizzare il modo hardware step (riferitevi alla Lezione 10).

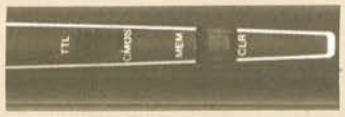
I) Ponete ora la sonda su una delle linee del bus dei dati e premete il tasto [HDWR STEP]. Osservate che la sonda può lampeggiare anche quando non si verificano cambiamenti di stati logici. Questo si verifica perché, ad ogni ciclo macchina, sul bus dei dati sono presenti (in multiplex) anche le informazioni relative al bus degli indirizzi. Il prossimo indirizzo può così essere memorizzato da IC2 quando il segnale $\overline{ALE}$ si porta alto (riferitevi alla Lezione 10).

J) Mettete la sonda sul punto IC2-11 (ingresso di clock del latch degli indirizzi). Quando viene premuto il tasto [HDWR STEP] controllate che sul segnale Address Latch Enable (Abilitazione della Memorizzazione Indirizzo, $\overline{ALE}$) si verifichi un impulso.

(continuazione)

IV. Memorizzazione degli impulsi

La possibilità della sonda logica di memorizzare un impulso può essere utilizzata per catturare uno o più impulsi (anche disturbi) che si verifichino in tempi tali in cui non sia per voi conveniente stare ad osservare il puntale della sonda. Tale caratteristica viene utilizzata quando si vuol vedere un impulso che si verifica solo saltuariamente o quando è difficile osservare il punto sotto esame.



Selettore della tensione di soglia e indicatore luminoso della memoria sulla sonda logica

A) Mentre siete ancora nel modo hardware step ponete la sonda su IC10-5 e premete il pulsante MEM CLR della sonda logica. Osservate che la luce di memoria si spegne. La memoria d'impulso (pulse memory) è stata azzerata. Ponete la punta della sonda sul piedino 5.

B) Osservate l'indicatore luminoso MEM quando premete il tasto [HDWR STEP]. L'indicatore si accende per segnalare che è stato individuato un impulso. Ulteriori impulsi non avrebbero alcun effetto. Quando usate la memoria, assicuratevi che la punta della sonda rimanga costantemente in contatto con il nodo da esaminare, in modo che la memoria non venga attivata dal rumore generato dal contatto elettrico.

RIASSUNTO

La sonda logica è uno strumento completo, facile da usare e indicato per analizzare i nodi logici. Potete usare la sonda per verificare discontinuità logiche e flussi di segnali, decodifiche di indirizzi, clock, interruttori e il funzionamento dei dispositivi presenti sul bus. Potete anche usarla per controllare le caratteristiche di funzionamento del Microprocessor Lab quando si trova nel modo hardware step.

Il generatore di impulsi logici

INTRODUZIONE

In questo esperimento userete il generatore di impulsi logici (Logic Pulser) per introdurre dei segnali nel μLab. Il generatore di impulsi HP 546A è un dispositivo che serve a stimolare il circuito sotto esame, e genera automaticamente impulsi di polarità, ampiezza, corrente, e durate prefissate, tali da poter pilotare i nodi dallo stato logico basso a quello alto e viceversa. È anche possibile scegliere se generare un solo pacchetto di impulsi (Pulse Burst Mode) o se generare una sequenza continua (Pulse Stream Mode).

PROCEDIMENTO

I. Utilizzo del generatore di impulsi logici

A) Collegate i cavetti di alimentazione del generatore di impulsi ai fori di alimentazione presenti sul μLab (lasciate collegata anche l'alimentazione della sonda logica).

B) Per generare un solo impulso, premete e rilasciate il pulsante posto sul generatore. Osservate che la lampadina posta sul puntale lampeggia una sola volta.

C) Tenete premuto il pulsante del generatore. Verificate come la luce del puntale lampeggi una volta e poi dopo circa un secondo inizi a lampeggiare rapidamente. Il generatore è così «programmato» per emettere impulsi ad una frequenza di 100 Hz e può inviare quindi brevi impulsi, ad elevata energia, ad una frequenza di 100 Hz. Nella Figura 16-7 è riportata questa forma d'onda dal duty-cycle (ciclo di lavoro) basso. Avendo un duty-cycle basso, si può limitare l'energia media inviata ad un IC ad un livello di sicurezza tale da prevenire eventuali danni.

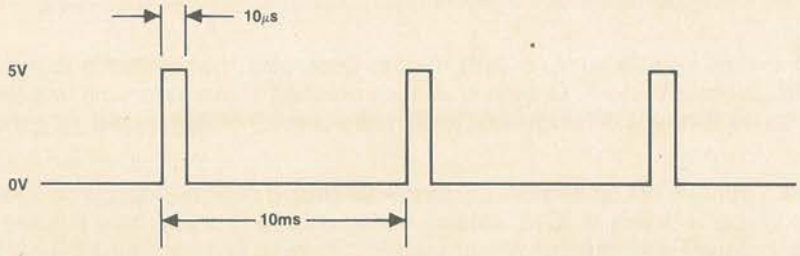


Figura 16-7. Forma d'onda che esce da un generatore di impulsi logici 546A nel caso di un nodo normalmente basso

D) Spostate verso il puntale l'interruttore in modo da bloccarlo. Notate che, sebbene il vostro dito non sia più sull'interruttore, la luce del generatore continua a lampeggiare.

E) Leggete l'etichetta sul generatore. Premendo il pulsante in rapida successione per 2,3,4 o 5 volte e mantenendolo premuto si possono programmare altri modi di uscita. Notate che i modi 10 o 100 burst inviano delle rapide sequenze di impulsi, per poi effettuare una pausa prima di inviare un'ulteriore sequenza. In tutti questi modi di funzionamento, lo strumento può essere bloccato semplicemente spostando in avanti l'interruttore.

Codici di programmazione per il generatore di impulsi logici

ESPERIMENTO 16-2

(continuazione)

II. Come iniettare impulsi in un nodo

A) Toccate, con il puntale del generatore d'impulsi, il punto IC18-9 (uscita del buffer dell'altoparlante) e programmatelo per il modo a 100 Hz (Figura 16-8). Ascoltate attentamente in modo da udire lo «scoppiettio» dei 100 Hz. Il generatore sta forzando (nello stato logico opposto) l'uscita del piedino 9 e iniettando una sequenza di impulsi nell'altoparlante.

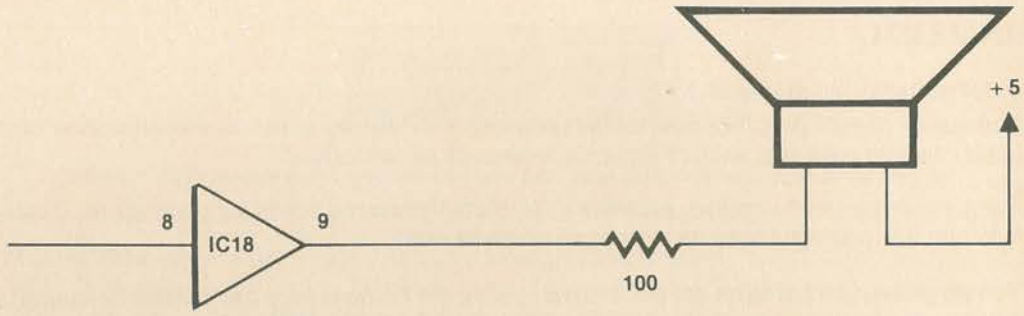


Figura 16-8. Circuito di pilotaggio dell'altoparlante del μLab

B) Toccate questo stesso punto con la sonda logica e verificate che questo piedino d'uscita è effettivamente forzato dal generatore di impulsi.

C) Cercate di inviare impulsi su V_{cc} o sulla massa. Osservate, con la sonda logica, che il generatore non può forzare questi nodi. Questo si verifica perché l'alimentatore ha una impedenza di uscita molto più bassa di quella dei dispositivi logici che possono essere forzati dal generatore.

D) Portate ora il puntale del generatore sul terminale destro della resistenza da 100 ohm, posta appena sotto e un po' a destra di IC18. Questo collegamento fa aumentare il livello del suono poiché ora vi siete collegati direttamente all'altoparlante. Adesso programmate il generatore per altri modi di uscita e ascoltate i differenti suoni.

III. Abilitazione di un dispositivo

A) Se sul display non è visualizzato uLAbUP, premete [RESET], in modo da assicurarvi che il programma monitor stia girando.

B) Portatevi con la sonda logica (Figura 16-9) sull'ingresso di clock del latch della porta di uscita (IC15-11). In questo momento non viene segnalata alcuna attività, indicando così che il programma monitor non sta indirizzando la porta d'uscita.

C) Servendovi del generatore di impulsi inviate degli impulsi a questo piedino. Osservate i LED di uscita del μLab. Provate ancora per qualche volta. Da ultimo provate ad utilizzare altri modi di uscita del generatore.

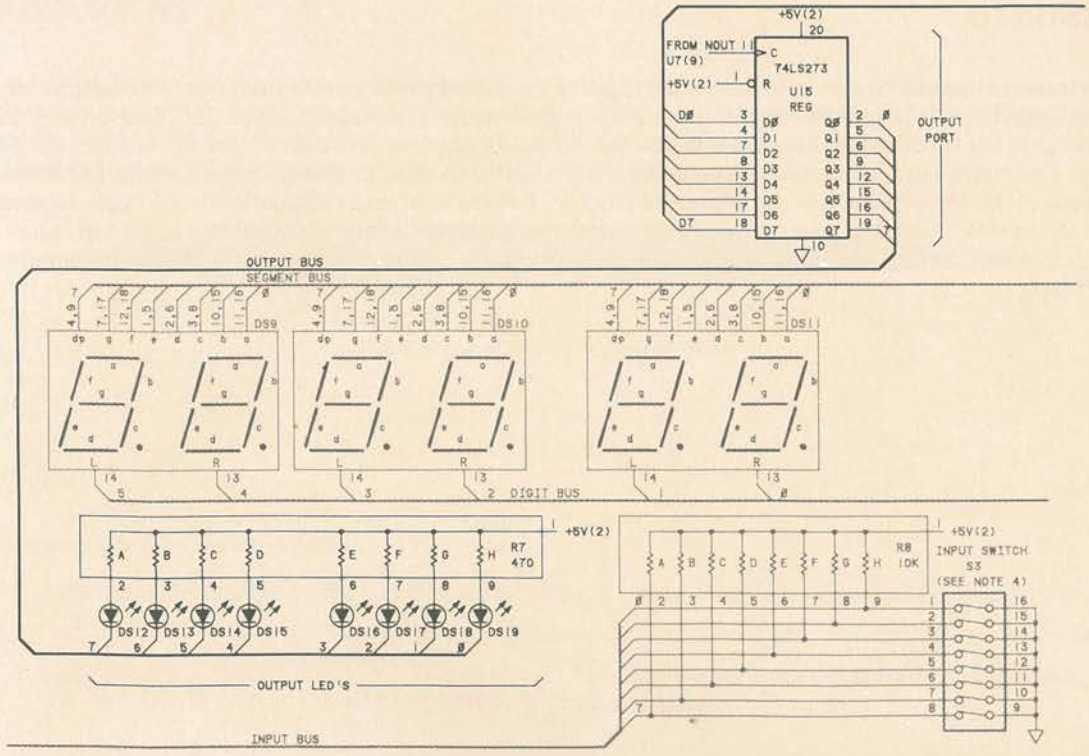


Figura 16-9. Circuito latch della porta d'uscita

VERIFICA

Perché i LED si sono comportati in modo così casuale? Che cosa avete osservato?

- Dallo schema, potete osservare che gli ingressi del latch di uscita IC15 sono tutti connessi al bus dei dati. Le uscite vanno ai LED.
- Dal momento che sta girando il programma monitor, c'è attività sul bus dei dati. Sul bus dei dati sono presenti configurazioni casuali di bit, quando questo è campionato in modo asincrono dai segnali provenienti dal generatore di impulsi.
- Il generatore fa sì che il latch memorizzi i dati provenienti dal bus dei dati, non tenendo conto dei tipi di dati presenti sul bus. Esso ignora il bus degli indirizzi e i segnali di controllo.
- Il fatto che questi dati vengano memorizzati vi dice che IC15 può immagazzinare dei dati quando riceve un segnale di clock. Perciò il suo ingresso di clock (piedino 11) è funzionante.
- Potete anche verificare che tutti e otto i bit del latch possono memorizzare e portare in uscita entrambi gli stati logici. Inviate, per un certo numero di volte, un impulso al piedino 11 e osservate come ciascun LED cambi di stato. Non avendo a disposizione dei LED di uscita per osservare questo fenomeno, si sarebbe potuto usare, al loro posto, la sonda logica.
- Avete potuto verificare che, con buona probabilità, IC15 non è rotto. Se non avesse funzionato nel modo corretto (attività presente sulle linee del bus dei dati, ma assente sul piedino 11), avreste potuto dedurre, con una certa sicurezza, di essere in presenza di un guasto.

ESPERIMENTO 16-2

(continuazione)

RIASSUNTO

Il generatore di impulsi forza impulsi nei nodi logici. Può essere programmato per generare singoli impulsi, sequenze continue o pacchetti distinti di impulsi. Il generatore può essere usato per forzare l'abilitazione degli integrati o per fornire un segnale di clock agli stessi. Si possono inviare impulsi agli ingressi dei circuiti logici ed osservare il loro effetto sulle uscite del circuito sotto esame. In questo esperimento, per esaminare i vari modi di uscita degli impulsi del generatore logico, è stato utilizzato l'altoparlante del μLab. Il generatore è stato anche utilizzato per memorizzare, in modo casuale, il bus dei dati del μLab nel latch della porta di uscita. Tramite i LED posti sulla porta di uscita è stato verificato il corretto funzionamento del latch stesso.

ESPERIMENTO 16-3

Il collaudo stimolo-risposta con una sonda e un generatore di impulsi

INTRODUZIONE

Questo esperimento mostra come il generatore possa stimolare i circuiti del μLab e come unitamente alla sonda logica, possa quindi rendere possibile un collaudo del tipo stimolo-risposta (Stimulus-Response). Un generatore logico inietta in un nodo un segnale di stimolazione e la sonda logica rileva la risposta negli altri nodi del circuito presenti lungo il percorso di propagazione del segnale.

Generatore di impulsi logici HP 546A

PROCEDIMENTO

I. Come seguire un flusso logico

A) Azzerate la memoria del μLab spegnendolo e riaccendendolo. Richiamate il primo indirizzo della RAM (0800) premendo [FETCH PC]. Premete [HDWR STEP] per entrare nel modo hardware step a partire dall'indirizzo 0800. Verificate che i LED del bus leggano dalla RAM, all'indirizzo 0800, il dato 00. Siccome il sistema è fermo, il display è spento.

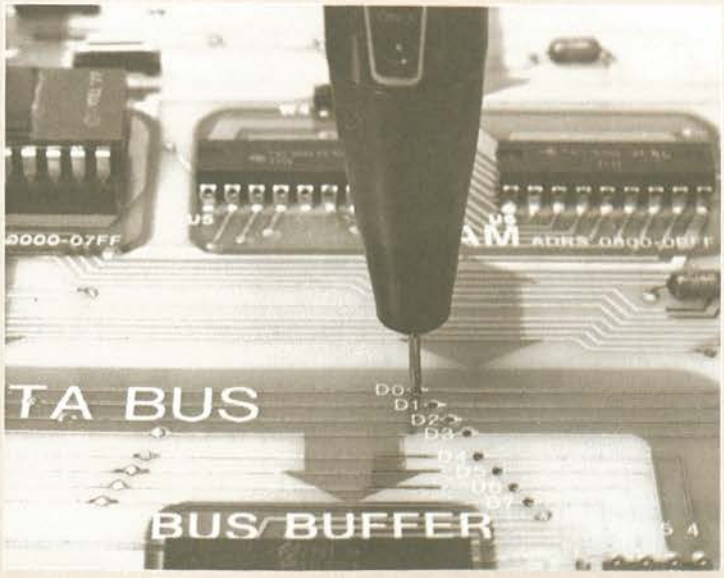


Figura 16-10. Sonda logica inserita nel foro di prova D0

(continuazione)

B) Utilizzando la sonda logica verificate che, sulla linea D0 del bus dei dati e sulla linea di abilitazione della RAM (IC5-8), il livello logico sia 0. Questo vi dice che la RAM è abilitata e che il dato presente sull'uscita D0 è 0. Adesso inserite la sonda nel foro di prova D0 (Figura 16-10) e lasciatevela.

C) Con il generatore d'impulsi iniettate un impulso in D0. Osservate che la sonda lampeggia una sola volta, indicando che il generatore ha forzato la linea del bus dei dati iniettando un impulso logico 1. Questo vi dimostra come è semplice iniettare un impulso in un nodo.

D) Iniettate degli impulsi nel piedino di abilitazione RAM (IC5-8) e osservate la sonda. Questa lampeggia, dal momento che il chip di RAM è momentaneamente disabilitato dall'impulso logico 1 iniettato dal generatore. Le uscite delle RAM sono spente e il bus dei dati è richiamato al livello logico 1 tramite le resistenze di richiamo (Pull-Up) da 10 K (poste proprio sopra gli interruttori del bus dei dati). Avete dimostrato che il piedino di abilitazione delle RAM può far sì che l'uscita D0 sia disabilitata.

E) Osservate (Figura 16-1) che il segnale $\overline{\text{RAM}}$ del piedino IC5-8 proviene da IC11-3 (uscita di un dispositivo OR). Utilizzate la sonda logica per verificare che siano bassi (0 logico) gli ingressi di tale dispositivo (IC11-1 e 2) e la sua uscita (IC11-3).

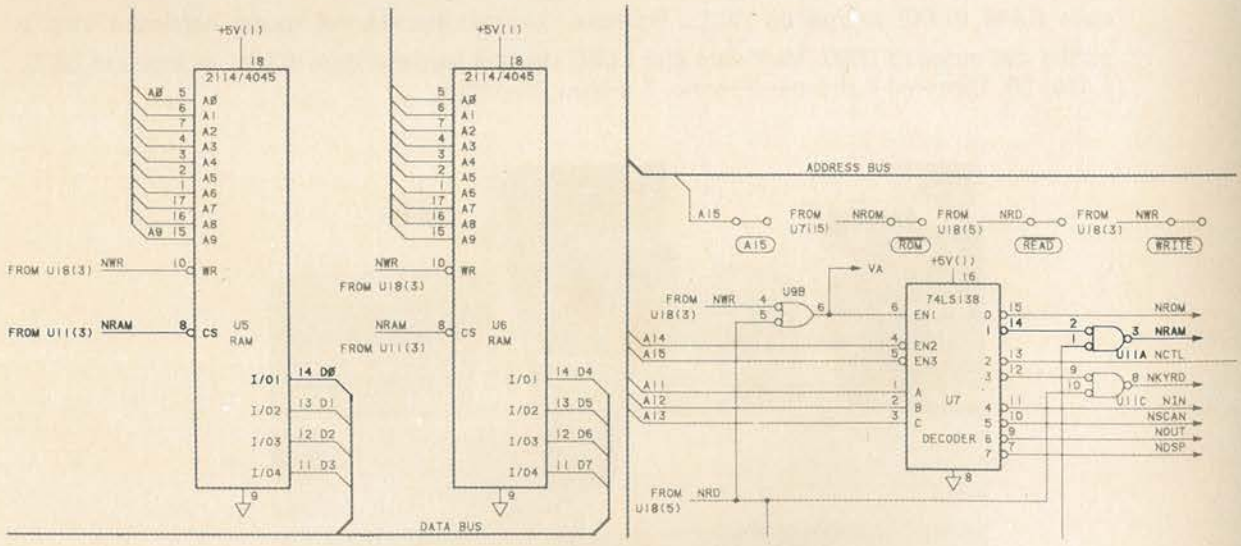


Figura 16-11. Circuito di decodifica della RAM

F) Per verificare il corretto funzionamento del dispositivo OR, inviate degli impulsi su uno e poi sull'altro dei suoi ingressi (piedini 1 e 2), dopo aver posto la sonda sull'uscita dello stesso (IC11-3). In uscita apparirà un impulso quando, su uno dei piedini di ingresso, viene inviato un impulso (1 logico).

G) Ritornate con la sonda sul foro di prova D0 del bus dei dati e osservate che gli impulsi inviati sugli ingressi del dispositivo OR (IC11-1 e 2) si propagano, attraverso il dispositivo stesso, al piedino 3. L'impulso prosegue poi verso il piedino di abilitazione della RAM (IC5-8), attraverso la RAM (IC5), e giunge finalmente all'uscita della RAM sulla linea D0 (IC5-14).

H) Ripercorrete, all'indietro, il percorso del segnale, da IC11-2 agli ingressi del decoder degli indirizzi (IC7). Iniettate degli impulsi su tali ingressi, uno per volta, e osservate che tutti i piedini di ingresso 2,3,4,5 e 6 fanno pulsare D0. Notate che lo stato di IC7-1 (che è collegato ad A11) non influisce su D0. Tutti questi ingressi fanno sì che sia vera un'altra uscita del decoder (ingressi di selezione, piedini 1, 2 e 3) o che tutte le uscite siano disabilitate (ingressi di abilitazione, piedini 4,5 e 6). La ragione, per cui un impulso su 1C7-1 non altera lo stato delle linee D0 del bus, sta nel fatto che la ROM è abilitata quando la linea A11 si porta bassa e che il dato presente su D0, proveniente dalla ROM per questo particolare indirizzo (0000), è pure 0.

II. Come generare pacchetti di impulsi (pulse burst)

Il generatore di impulsi logici può generare pacchetti di 10 o 100 impulsi. Questo modo di operare è utile quando occorra avere un preciso numero di impulsi di clock al fine di definire lo stato di un contatore, di un registro di shift o di qualche altro circuito sequenziale.

A) Accendete e spegnete il μLab onde azzerare la memoria. Premete [FETCH PC] per posizionare l'indirizzo a 0800 (il primo indirizzo di RAM). Premete [HDWR STEP]. Osservate che i LED del bus degli indirizzi segnalano appunto l'indirizzo 0800.

B) Individuate l'ingresso di clock sul flip-flop di single step (IC10-3). La Figura 16-12 mostra che questo piedino è connesso ad un circuito di antirimbalzo, che è a sua volta collegato al tasto [HDWR STEP]. Premete, per alcune volte, il tasto [HDWR STEP] e osservate che i LED sul bus degli indirizzi vengono incrementati.

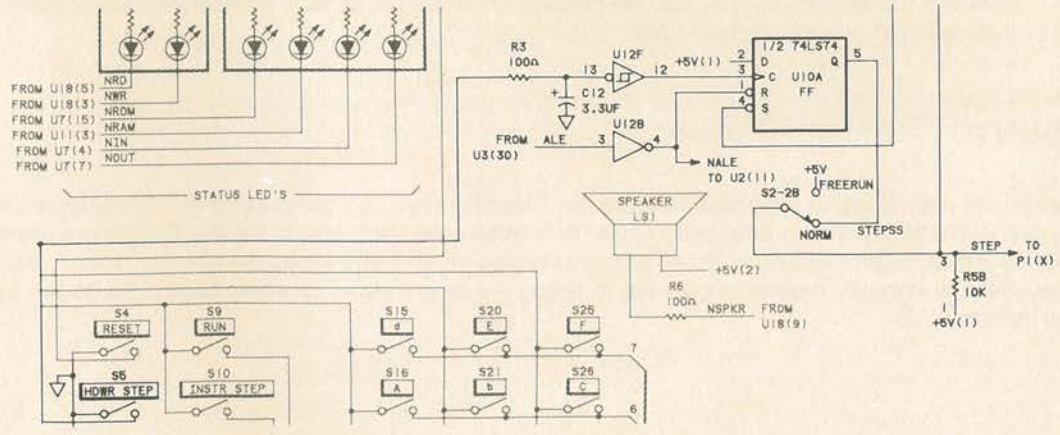


Figura 16-12. Generatore di impulsi logici utilizzato per incrementare il circuito single-step

ESPERIMENTO 16-3

(continuazione)

C) Tramite il generatore di impulsi, iniettate alcuni impulsi nell'ingresso di clock (IC10-3) e osservate come i valori indicati dai LED si incrementino ancora.

D) Programmate il generatore per i modi 1 Hz e 10 Hz e ripetete il punto C. I LED del bus degli indirizzi si incrementano alla frequenza di uscita degli impulsi.

E) Programmate il generatore per il modo 100 Hz e ripetete il punto C. Osservate che, mentre l'indirizzo passa da 0800 a OAEE, i LED del bus dei dati sono spenti. Durante questa sequenza il μLab esegue delle istruzioni di NOP (00) che sono memorizzate in RAM, prima dell'indirizzo OAEE, dal programma che viene eseguito all'accensione del μLab. Dopo che il μLab ha raggiunto l'indirizzo OAEE, inizia ad eseguire un programma ed i LED del bus si accendono ora (a caso, apparentemente) seguendo l'attività sul bus. Il bus è attivo poiché, quando viene eseguito un hardware step, viene eseguita un'istruzione (o parte di essa). Durante il periodo di tempo in cui vengono eseguiti circa 1000 NOP, non si verifica nessun salto di indirizzo o trasferimento dati. Negli indirizzi superiori a OAEE sono invece memorizzati programmi e dati anziché zeri.

F) Premete [RESET] per ridare il controllo al monitor. Portatevi sull'indirizzo 0800. Per ritornare alla prima istruzione NOP in RAM, premete [HDWR STEP].

G) Mettete il generatore di impulsi su IC10-3. Programmate dapprima l'invio di 2 pacchetti da 100 impulsi. Programmate poi l'invio di 5 pacchetti da 10 impulsi. Inviate infine sei impulsi singoli. Avete così inviato un totale di 256 impulsi, e i LED del bus degli indirizzi dovrebbero indicare l'indirizzo 0900 (0900 hex meno 0800 hex = 256 decim.).

H) Programmate alcuni altri conteggi di impulsi (sono interessanti i valori 511 e 361). Ripetete il punto F, quando l'indirizzo della RAM supera OAEE. Ricordate che ci sono meno di 1024 locazioni di RAM vuote (00), per cui non cercate di inviare più di 1024 impulsi successivi aspettandovi di vedere qualcosa di prevedibile.

RIASSUNTO

Agli ingressi dei circuiti logici sono stati iniettati degli impulsi tramite il generatore di impulsi logici, mentre contemporaneamente venivano analizzate le uscite facendo uso della sonda logica. È stata poi esaminata, con la sonda logica, la propagazione dei segnali attraverso gli elementi logici del circuito. Infine, sono stati inviati pacchetti di impulsi, mentre si operava in modo hardware step, osservandone l'effetto sui LED del bus degli indirizzi.

Il rivelatore di corrente

INTRODUZIONE

Questo esperimento mostra come sia possibile usare il generatore di impulsi unitamente ad un rivelatore di corrente (Current Tracer). Il rivelatore di corrente HP 547A rivela l'attività della corrente in un nodo per mezzo di un trasduttore induttivo posto alla sua estremità. Quando il rivelatore è posizionato su una linea di un segnale logico pulsante, regolando il suo controllo di sensibilità e osservando l'intensità della luce di indicazione potete identificare i percorsi della corrente e i suoi valori, localizzando in questo modo una malfunzione in un dispositivo posto su di un nodo. In questo esperimento, per osservare l'attività della corrente nel μLab, userete il rivelatore di corrente sia da solo che in tandem con il generatore d'impulsi.

Le possibilità, che il rivelatore di corrente vi offre, per risolvere il problema della ricerca guasti, sono meno ovvie di quelle fornite dalla sonda logica o dal generatore di impulsi. Questo è vero perché il rivelatore di corrente analizza correnti e non tensioni, grandezza con cui avete ormai acquisito familiarità. Tuttavia, una volta che avrete raggiunto una certa confidenza nel suo uso (cosa peraltro non troppo difficile), potete trarre da questo strumento numerosi benefici. Esperienze frustranti di ricerca di guasti, come si verifica quando si hanno dei nodi «bloccati» (Stuck) o un corto circuito tra linee di alimentazione, possono essere affrontate e risolte in modo diretto ed efficace. In precedenza, magari, avreste cercato d'identificare questo tipo di malfunzioni interrompendo delle piste del circuito stampato, tagliando dei piedini o chissà cos'altro.

Il rivelatore di corrente HP 547A

PROCEDIMENTO

I. Uso del rivelatore di corrente

A) Con la sonda logica ed il generatore di impulsi ancora connessi all'alimentazione, collegate i fili di alimentazione del rivelatore di corrente ai fori di alimentazione del μLab.

B) Se il display non mostra già uLAb UP, premete [RESET].

C) Affinché il rivelatore di corrente possa operare in modo corretto, il suo puntale deve essere opportunamente posizionato sulla pista del circuito che si vuole esaminare. Il puntale del rivelatore dovrebbe sempre essere perpendicolare alla scheda e avere i piccoli fori, posti lateralmente alla punta, allineati con (un foro davanti e uno dietro) il percorso della corrente (Figura 16-13).

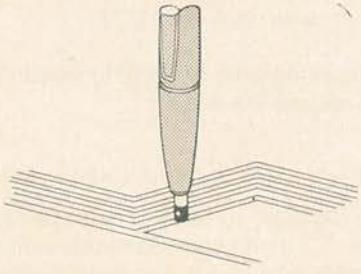


Figura 16-13. Orientazione corretta del rivelatore di corrente 547A rispetto alla traccia del circuito

(continuazione)

D) Posizionate il puntale del rivelatore di corrente su una qualunque delle linee del bus degli indirizzi, poste sulla parte alta della scheda. Dal momento che la punta è isolata, potete porla direttamente in contatto con i conduttori elettrici. Assicuratevi che la punta sia orientata correttamente.

E) Ruotate la piccola manopola della sensibilità e osservate che la lampadina vicino alla punta del rivelatore cambia di intensità.

F) Tenendo ancora in contatto la punta con una linea del bus degli indirizzi, regolate opportunamente la sensibilità in modo che la lampadina abbia un'intensità pari circa alla metà del valore massimo.

G) Adesso, senza modificare l'intensità, provate le altre linee del bus degli indirizzi e verificate che tutte diano circa la stessa intensità. Il rivelatore di corrente sta rivelando principalmente la corrente che va dai circuiti di pilotaggio del bus degli indirizzi IC1 e IC2 ai LED del bus degli indirizzi.

H) Ruotate la punta del rivelatore posto su una linea di indirizzo e osservate come la lampadina diventi meno luminosa. La lampadina si offusca perché la corrente fluisce lungo le piste e non in senso perpendicolare ad esse. Questo dovrebbe mostrare quanto è importante mantenere il puntale della sonda orientato in modo corretto, quando si vuole rivelata una corrente lungo una pista.

I) Riferendovi alla Figura 16-14, esaminate altre piste della scheda. Notate che, quando è posta sulle piste dell'alimentazione e del display, la lampadina si illumina molto (su queste piste si verificano notevoli «sbalzi» di corrente generati da molti dei circuiti integrati). Le linee in cui non si ha attività non danno alcuna indicazione luminosa.

J) La regolazione della sensibilità permette al rivelatore di corrente di rilevare correnti comprese tra 1 mA e più di 1 A. Regolate questo controllo in modo da poter confrontare i livelli di corrente sulle varie linee della scheda e, riferendovi allo schema, guardate se questi concordano con quello che vi potreste aspettare. Consiglio: è più facile rilevare piccole differenze in corrente se la sensibilità è predisposta per dare un'indicazione luminosa molto tenue.

II. Rilevazione di impulsi di corrente

A) Accendete e spegnete il μLab per azzerare la memoria.

B) Regolate il rivelatore di corrente in modo che, quando lo stesso sia posto su una linea di indirizzo, la lampadina sia ben luminosa (sensibilità elevata).

C) Premete [FETCH PC] e [HDWR STEP]. Osservate l'attività della corrente (come varia cioè la corrente) sulle linee degli indirizzi e su tutte le altre linee logiche, sui piedini di alimentazione del microprocessore e su alcune delle piste di alimentazione. L'attività che osservate indica che il microprocessore sta ancora operando al suo interno, aspettando che la sua linea di Ready venga rilasciata dal circuito hardware step.

D) Mentre state rilevando la corrente sulla linea di indirizzo A0 (posizionatevi tra il foro per la sonda, posto sul bus degli indirizzi, e il LED relativo alla stessa linea) premete più volte [HDWR STEP].

E) Spostatevi sulla linea A1 e osservate che essa lampeggia ogni volta che premete il tasto [HDWR STEP] (in corrispondenza con il cambiamento dello stato del LED A1). Questo lampeggiare ci indica due delle caratteristiche del rivelatore di corrente:

- Risponde solamente a variazioni di corrente (non alla corrente continua).
- Ha una capacità di allungare gli impulsi di corrente simile a quello della sonda logica.

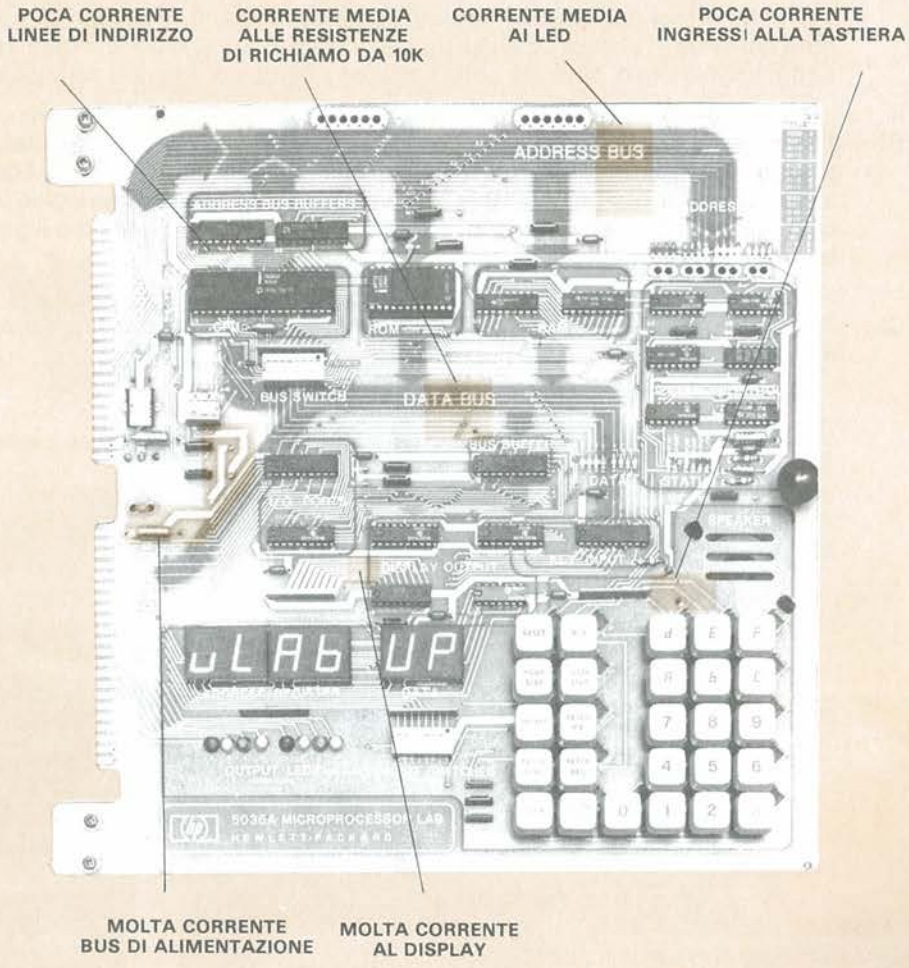


Figura 16-14. Attività della corrente per diversi circuiti del μLab

(continuazione)

III. Uso del generatore di impulsi e del rivelatore di corrente nei collaudi stimolo-risposta

Il generatore di impulsi può essere utilizzato per introdurre un'attività impulsiva in un nodo non attivo. Il rivelatore di corrente può poi essere utilizzato per seguire la corrente che fluisce da questo nodo.

A) Osservate che tutte le linee del bus dei dati sono nello stato logico 0 (i LED del bus dei dati sono tutti spenti). Potete dedurre che almeno un dispositivo del bus è abilitato. Quando non sono presenti sul bus dei dispositivi abilitati, le linee del bus sono infatti richiamate verso l'alto da resistenze da 10 K poste sul bus dei dati stesso. (Anche baffi di stagno o altri cortocircuiti tra la massa e le linee del bus dei dati potrebbero portare a questa situazione).

Dal momento che il microprocessore è fermo (nel modo hardware step), sulle linee del bus dei dati sono presenti solamente correnti continue, costanti. Il rivelatore di corrente non rileva correnti continue. Tuttavia se si usa un generatore di impulsi per iniettare impulsi di corrente in un nodo, il rivelatore di corrente potrà seguirne il percorso. Provate con una linea del bus dei dati.

B) Ponete il puntale del generatore di impulsi nel foro di prova D0 posto sul bus dei dati. Programmatelo nel modo 100 Hz (premete e spostate in avanti l'interruttore). Adesso nel nodo vengono iniettati impulsi con una corrente sufficiente a pilotare (a cambiare cioè lo stato logico dei) i circuiti posti sulla linea D0. Utilizzate la sonda logica per verificare che il generatore di impulsi stia effettuando veramente tale operazione.

C) Ponete il puntale del rivelatore di corrente sul puntale del generatore di impulsi in modo da determinare la corrente che entra nel nodo (Figura 16-15).

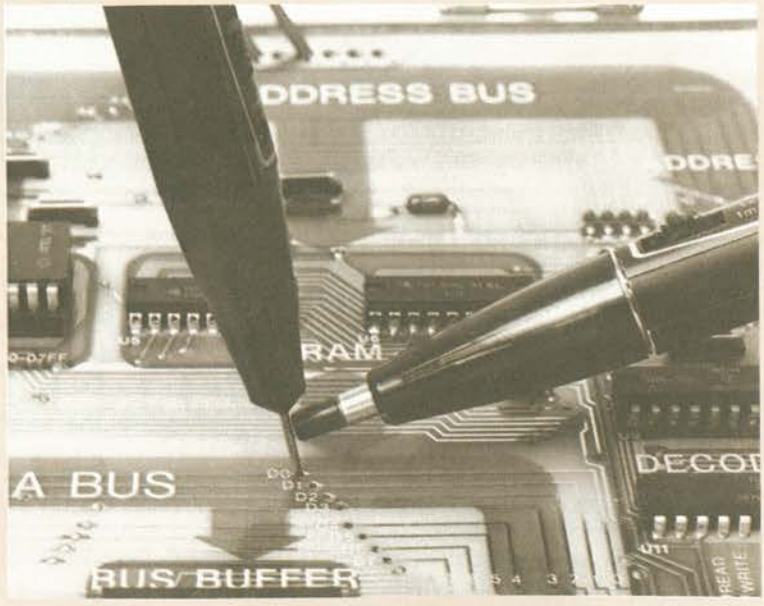


Figura 16-15. Misura con un rivelatore di corrente della corrente iniettata dal generatore di impulsi in un nodo

D) Regolate la sensibilità in modo che la lampada del rivelatore sia appena illuminata. Controllate che il puntale del rivelatore di corrente sia tenuto allineato con il puntale del generatore di impulsi. La manopola di regolazione si troverà così circa a metà della rotazione possibile. Questa posizione significa che la corrente si trova probabilmente tra 10-100 mA, valore superiore a quello dovuto alle resistenze di richiamo da 10 K o agli ingressi dei dispositivi logici, ma inferiore a quello che si avrebbe in caso di un cortocircuito verso massa, dal momento che il generatore di impulsi può fornire fino a 1 A di corrente nel caso di cortocircuiti verso massa o verso V_{cc}. Quindi più o meno ci si trova nel campo della corrente che serve a forzare l'uscita di un MOS o di un TTL. Il confronto con correnti di circuiti conosciuti e l'esperienza acquisita con il rivelatore di corrente vi permetteranno di avanzare ipotesi ragionevoli come queste.

E) Adesso, senza modificare la sensibilità, seguite sulla scheda questa corrente. Controllate le due direzioni (sinistra e destra) della pista D0 per vedere in quale direzione la corrente si muove. Notate che non sembra vada da nessuna parte. Dal momento che la corrente non può scomparire, è lecito supporre che sia finita tutta sull'altro lato della scheda. Tirate in modo deciso la manopola nera posta a metà del bordo destro della scheda, in modo da sbloccare la scheda stessa. Ruotate la scheda attorno alle cerniere e controllate la corrente sul lato inferiore della piastra vicino al foro di prova D0 (Figura 16-16). Ora dovreste essere in grado di seguire la corrente lungo la pista fino a IC5-14. Potete in effetti seguire la corrente proprio fino a tale piedino.

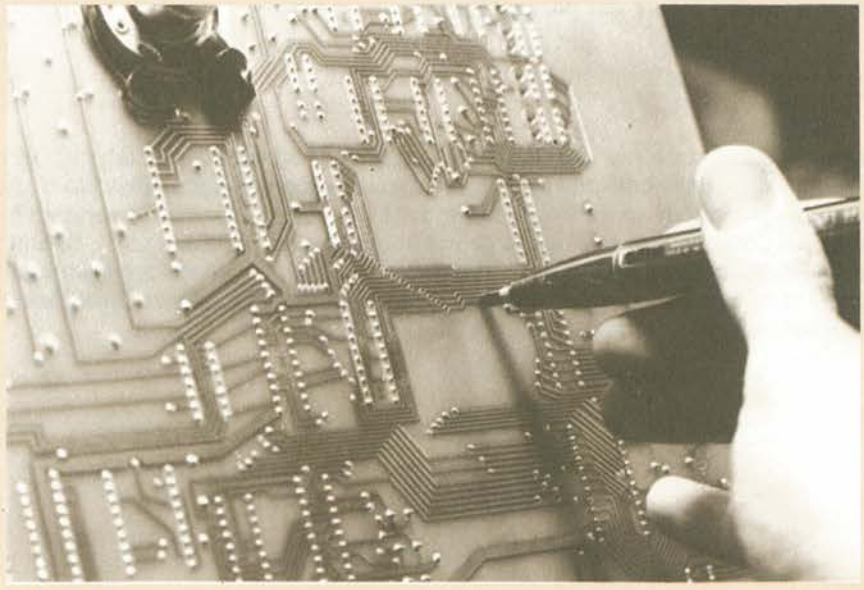


Figura 16-16. Come seguire la corrente sul retro della scheda servendosi del rivelatore di corrente

Dal momento che potete vedere, dai LED di stato, che il μLab sta eseguendo un'operazione READ (lettura) della RAM, il fatto che la corrente segua questo percorso non dovrebbe sorprendervi. Se la RAM non fosse stata abilitata, questo particolare percorso della corrente avrebbe indicato una uscita guasta della RAM. Con altri cinque dispositivi collegati a questo nodo, siete in grado di determinare quale dei sei è attivo. Se questo piedino della RAM fosse stato bloccato, il

(continuazione)

rivelatore di corrente vi avrebbe condotto ad esso. È difficile isolare i dispositivi posti, come nel nostro caso, su un bus servendosi solo di strumenti in grado di rilevare la tensione.

F) Spegnete il generatore di impulsi, spostando indietro l'interruttore. Premete [RESET] in modo da ridare il controllo al monitor. Premete [FETCH ADRS] [0] [0] [0] [0]. Premete [HDWR STEP].

G) Predisponete il generatore di impulsi nel modo 100 Hz e seguite il percorso della corrente lungo la linea D0. (Riferitevi ai punti da B a D se avete dimenticato tale procedimento). Dovreste arrivare alla ROM, IC4-9 (perché adesso è abilitata la ROM). Cercate di seguire il percorso della corrente su altre linee di dati.

H) Adesso posizionate il generatore sull'alimentazione V_{cc} o su una pista di massa e seguite la corrente. La prima differenza che noterete riguarda l'ampiezza della corrente fornita dal generatore di impulsi. La lampada del rivelatore di corrente è completamente accesa anche con una sensibilità minima (1 A). Un'altra differenza è che il percorso della corrente riporta sempre al cavo di massa del generatore di impulsi, dal momento che la corrente di uscita del generatore fluisce attraverso questo filo. Quando si verificano cortocircuiti di alimentazione sul bus, il generatore e il rivelatore possono essere utili per localizzarlo.

RIASSUNTO

Il rivelatore di corrente è utilizzato per identificare le cause dei nodi «bloccati» (stuck). Esso può dirvi, approssimativamente, il valore della corrente impulsiva e quale percorso segua. Quando si usa un generatore di impulsi per iniettare corrente in un nodo privo di attività impulsiva, è possibile valutare la sua impedenza e la natura generale del problema (p. es., uscita di un circuito logico, cortocircuito). Si può trovare quindi il vero punto a bassa impedenza seguendo il percorso della corrente dal generatore di impulsi a tale particolare punto del nodo. O la corrente va da qualche parte dove non dovrebbe andare (una linea in corto), o entra in un componente bloccato, cortocircuitato o acceso.

La sonda logica, il generatore d'impulsi logici e il rivelatore di corrente sono degli utili strumenti portatili che vengono usati, da soli o insieme, per ricercare dei guasti nei circuiti digitali. Tutti e tre possono essere alimentati direttamente dal circuito sotto esame.

La sonda logica indica lo stato logico di un nodo («0», «1» o un cattivo livello) e vi segnala quando vi è attività in un nodo. È particolarmente utile per identificare la presenza di nodi collegati male o il verificarsi di eventi singoli.

Il generatore di impulsi inietta impulsi in un nodo, forzando il circuito a delle risposte. Questi impulsi di risposta possono poi essere analizzati con la sonda logica, con il rivelatore di corrente o, più semplicemente, osservando il comportamento del circuito in esame.

Il rivelatore di corrente analizza l'attività della corrente sulle linee di segnale (piste del circuito, piedini dei componenti e fili). Il percorso della corrente in un nodo bloccato può così essere seguito fino al punto del guasto.

DOMANDE

Lezione 16

1. Quando il puntale di una sonda logica lampeggia rapidamente, vuol dire che il nodo logico in esame:
   a. è ad un cattivo livello logico.
   b. ha un'attività logica che presenta dei cambiamenti rapidi.
   c. ha un'attività logica instabile.
   d. è un nodo difettoso.

2. Nel caso venga rilevato un singolo impulso (livello logico 1) della durata di un microsecondo, la lampadina sul puntale della sonda logica:
   a. si accende per un istante, troppo velocemente per essere osservabile.
   b. si spegne per un istante, troppo velocemente per essere osservabile.
   c. si accende per un intervallo di tempo abbastanza lungo perché il fenomeno sia osservato.
   d. si accende e rimane accesa.

3. Gli impulsi di uscita del generatore di impulsi:
   a. sono troppo brevi per essere rilevati dalla sonda logica.
   b. possono danneggiare i dispositivi logici.
   c. possono solamente forzare a un valore logico basso i nodi alti.
   d. possono essere usati per forzare i nodi logici a livelli bassi o alti.

4. La quantità di corrente che esce dal generatore di impulsi:
   a. è sempre almeno 1 A.
   b. dipende dal fatto che venga usata la sonda logica o il rivelatore di corrente.
   c. dipende da quale modo di uscita degli impulsi viene selezionato.
   d. varia con l'impedenza del nodo pilotato.

5. Il generatore d'impulsi dovrebbe essere usato insieme al rivelatore di corrente, quando:
   a. non c'è attività di corrente sul nodo in prova.
   b. non c'è corrente sul nodo in prova.
   c. il nodo difettoso non è stato trovato.
   d. il nodo non è difettoso.

6. Quando si segue la corrente con il rivelatore di corrente, il controllo della sensibilità dovrebbe generalmente essere predisposto in modo che la lampadina:
   a. sia un poco meno luminosa del massimo.
   b. sia un poco più luminosa del minimo.
   c. si spenga appena.
   d. inizi a lampeggiare.

LEZIONE 17

Analizzatori di firma e analizzatori logici

INTRODUZIONE

Gli *Analizzatori di Firma* e gli *Analizzatori Logici* costituiscono due classi specializzate di strumenti di collaudo che sono usati nello sviluppo dei prodotti, nella produzione e nell'assistenza presso clienti. Possono essere anche molto utili nella ricerca dei guasti nei sistemi a microprocessori. Questa lezione descrive tali strumenti ed illustra come possono essere utilizzati per effettuare una ricerca guasti nei prodotti basati sui microprocessori.

ANALISI DI FIRMA

L'*Analisi di Firma* (SA-Signature Analysis) è una tecnica, estremamente precisa e di facile uso, utilizzata per identificare i nodi logici guasti. L'analizzatore di firma può convertire in numeri di 4 cifre, detti «firme» (Signature), le lunghe e complesse sequenze di dati seriali presenti nei nodi logici di un sistema a microprocessore. Queste firme ci dicono se il nodo è correttamente attivo.

L'insieme dei caratteri utilizzati per l'analisi di firma è costituito da 16 caratteri (0-9,A,C,F,H,P,U). Usando questo insieme di caratteri, viene eliminata la confusione che si verifica quando vengono rappresentati, su un display a sette-segmenti, i caratteri esadecimali b e B. Di conseguenza, tutti i caratteri sono facilmente distinguibili uno dall'altro.

In generale sono controllati con un analizzatore di firma i punti circostanti l'area in predicato di malfunzionamento (Figura 17-1), finché viene trovata una firma che non concorda con quella riportata nel manuale di assistenza. Per localizzare in modo esatto il guasto occorre allora ritornare lungo il percorso del segnale fino al punto in cui viene rilevata una firma corretta. Una volta che è stato individuato il nodo difettoso o diventa immediato il riconoscimento del componente guasto, oppure lo stesso può essere localizzato utilizzando un rilevatore di corrente o altre tecniche. Le firme dei nodi digitali possono essere utilizzate nello stesso modo in cui sono oggi utilizzate le informazioni di tensioni e forme d'onda riportate sugli schemi dei manuali di assistenza di circuiti analogici.

Le firme di un nodo sono significative perché sono generate con un programma di stimolazione di test contenuto nel circuito sotto esame o in un adattatore esterno. Questo programma di stimolazione esercita in modo controllato e ripetibile parti specifiche del circuito.

Potreste chiedervi come vengano raccolte dai costruttori le firme corrette. La procedura normale consiste nel prendere un prodotto, che si sa funzionante, e raccogliere da esso, nodo per nodo, le firme di riferimento. Si segue questa tecnica, perché è molto difficile riuscire a predire «a tavolino» l'enorme quantità d'informazioni, per di più complesse, contenute in ciascuna firma. La raccolta delle firme è un processo lungo, in cui si possono e in effetti si verificano errori (controllate sempre le errata corrige, che riportano, per i diversi prodotti, gli eventuali aggiornamenti).

Figura 17-1. Analizzatore di firma 5004A

Se a causa di una revisione del prodotto, viene cambiata anche una sola parola della ROM potrebbe essere necessario cambiare qualche o magari tutte le firme indicate nella documentazione. Assicuratevi perciò, che il numero di serie del prodotto che state utilizzando corrisponda alla documentazione in vostro possesso.

Non c'è cosa peggiore di una firma «quasi» giusta: una firma è solo giusta o sbagliata. A causa dell'algoritmo usato dall'analizzatore di firme, il valore 5H22 non è più «vicino» a 5H23 di quanto non lo sia a C7FP.

I segnali, richiesti dall'analizzatore di firma per generare una firma, sono: DATA, START, STOP e CLOCK. L'ingresso DATA riceve il dato dal nodo in prova. Il segnale START, generato dal prodotto in prova, comunica all'analizzatore di firma quando deve iniziare od acquisire il dato, mentre il segnale STOP serve ad indicare quando deve essere interrotta l'analisi. All'interno dei segnali START e STOP, il dato viene acquisito ogni volta che si verifica un nuovo impulso di CLOCK. Nella documentazione di manutenzione per il prodotto in prova devono essere esplicitamente specificati i punti di connessione per questi ingressi.

Negli esperimenti che seguono raccoglierete le firme relative a nodi del μLab, seguendo diversi modi di analisi di firma. Verrà poi discussa la relazione esistente tra le diverse firme e le corrispondenti sezioni circuitali.

TABELLE DI FIRMA PER IL MICROPROCESSOR LAB

Le tabelle di firma del μLab riportano le firme relative ad ogni piedino dei circuiti integrati e alle linee del bus dei dati e degli indirizzi. Queste tabelle sono riportate in Appendice C. Nella maggior parte dei casi sarà riportata una firma per ogni piedino; esistono però diversi modi in cui queste indicazioni possono essere date. Se il piedino è collegato direttamente a massa o +5V, la voce corrispondente

della tabella indicherà semplicemente GND o V_{cc} piuttosto che dare l'indicazione della firma. La firma di V_{cc} è riportata all'inizio della tabella, mentre la firma di GND è sempre 0000.
La tabella può anche riportare, come firma, 1 o 0, che hanno lo stesso significato di V_{cc} o GND: cambia solo il fatto che il segnale è in tal caso una uscita di una porta logica, non collegata direttamente a V_{cc} o GND. Per ultimo, la tabella può mostrare le firme V_{cc} o GND con un B che hanno lo stesso significato di 1 e 0: la luce posta sul puntale della sonda dell'analizzatore di firme dovrebbe però in questo caso lampeggiare. Il valore B significa che, sebbene quando arrivi il fronte del clock il segnale sia sempre allo stesso livello logico, negli altri istanti la linea si trova ad un diverso livello.

ESPERIMENTO 17-1

Funzione free-run del Microprocessor lab

INTRODUZIONE

Il modo di operare Free-Run (gira liberamente) richiede che solo una parte minima di un circuito (in questo caso il microprocessore) «eserciti» la maggior parte possibile del resto del circuito. Affinché questo si verifichi, non occorre che i dispositivi come ROM, RAM e I/O siano in funzione. Qualsiasi linea di reazione (ad esempio il bus dei dati) verso il microprocessore deve essere interrotta, in modo che la sua operatività non sia influenzata da segnali errati o imprevedibili. Questa tecnica serve a generare una stimolazione di prova per ricavare delle firme. Il vantaggio di usare questo modo di analisi di firma sta nel fatto che si richiede che solo una piccola parte del circuito operi correttamente.

PROCEDIMENTO

I. Preparazione del modo free-run

A) Muovete verso l'alto gli otto interruttori a slitta del bus dei dati (Figura 17-2). In questo modo è stato interrotto (aperto) il bus dei dati tra il microprocessore e il resto del sistema. Dal momento che non esistono più percorsi di ritorno dalla memoria al microprocessore, nessuna istruzione viene più inviata allo stesso, ed il microprocessore è così libero di funzionare ad «anello aperto».

B) Spostate verso l'alto l'interruttore di *Free-Run* (FR), che si trova in basso a sinistra rispetto agli interruttori del bus dei dati. Questo interruttore fa eseguire al microprocessore un'istruzione MOV A,A (codice 7F) ogni volta che viene eseguita un'operazione di lettura (Figura 17-3). L'istruzione MOV A,A non effettua alcuna operazione, analogamente a quanto si ha per l'istruzione NOP. Il microprocessore legge questa istruzione e incrementa quindi l'indirizzo per leggere l'istruzione successiva. Dal momento che il bus dei dati non è collegato, questa istruzione viene letta ad ogni indirizzo, ed il microprocessore è così obbligato ad eseguire in continuazione la stessa istruzione (e solo questa). Perciò le linee del bus degli indirizzi continuano ad incrementarsi passando per tutti i 2^{16} indirizzi possibili.

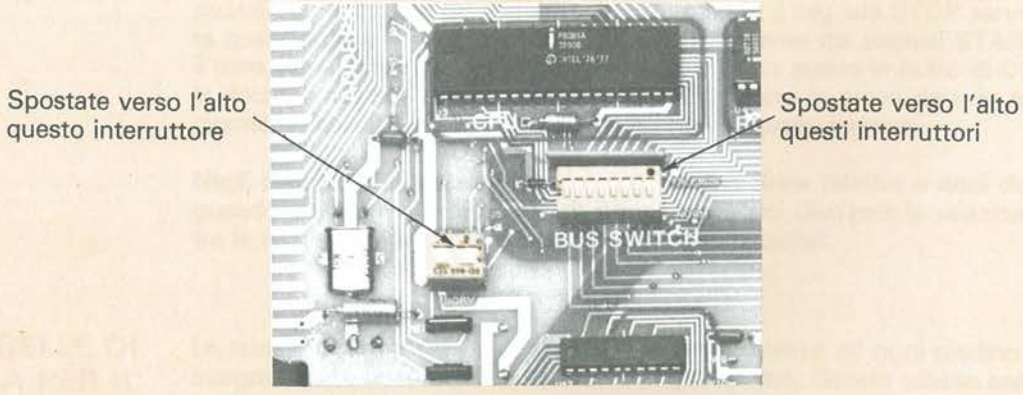


Figura 17-2. Posizione degli interruttori di test del μLab

Oltre al microprocessore, solamente pochi altri circuiti del μLab possono impedire che il modo free-run funzioni. È il caso dell'alimentatore, del quarzo, degli interruttori di prova e dei circuiti direttamente collegati ai piedini di controllo del microprocessore. In un sistema basato su microprocessore, l'insieme di questi circuiti, spesso conosciuti come *kernel* (cioè nocciolo), può essere considerato un punto di separazione fondamentale per localizzare dei guasti. Se il sistema non riesce ad operare in free-run, il problema viene ristretto infatti a questo piccolo numero di componenti.

C) Osservate gli indicatori LED del bus degli indirizzi. Il lampeggiare visibile dei LED A15 e A14 e la diminuzione di luminosità dei LED da A13 a A0 (lampeggiano troppo velocemente per essere visti) indicano che il μLab sta continuamente passando lungo tutto il campo di indirizzamento. Notate che il LED di stato «READ» è ben acceso. Questa luminosità è legata al fatto che il microprocessore sta continuamente leggendo l'istruzione MOV A,A (forzata dal diodo D1 e dalle resistenze di richiamo da 10K). I LED di stato contrassegnati con ROM, RAM, INPUT e OUTPUT, che sembrano lampeggiare tutti insieme, in realtà lampeggiano in rapida successione, man mano che gli indirizzi, incrementandosi, passano attraverso l'area di memoria assegnata ai diversi dispositivi.

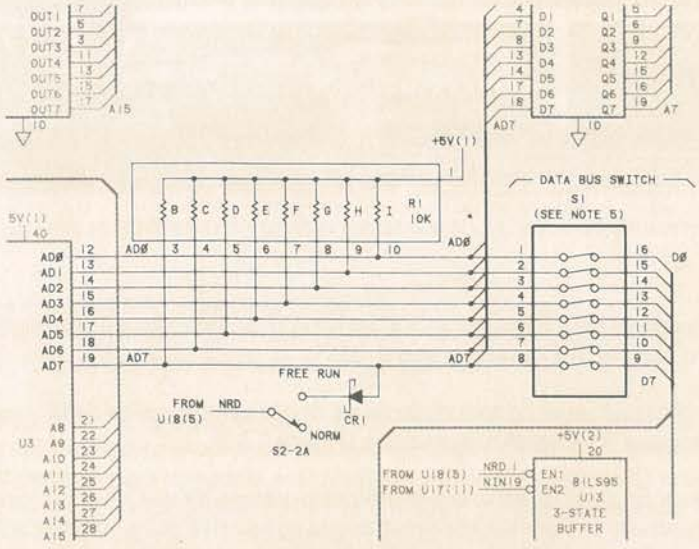


Figura 17-3. Circuito utilizzato per aprire il bus dei dati e far operare il μLab in free-run

II. Lettura delle firme

A) Collegate il cavetto GND dell'analizzatore al foro di massa del μLab (↓), i fili di START e STOP al foro A15 e il filo di CLOCK al foro $\overline{\text{READ}}$ (Figura 17-4).

B) Posizionate gli interruttori START, STOP e CLOCK, posti sull'analizzatore di firma, in modo che indichino il fronte di salita (interruttore rilasciato). L'interruttore GATE lampeggia, per mostrare che è presente una «finestra» per la raccolta della firma. La finestra è costituita dall'intervallo di

(continuazione)

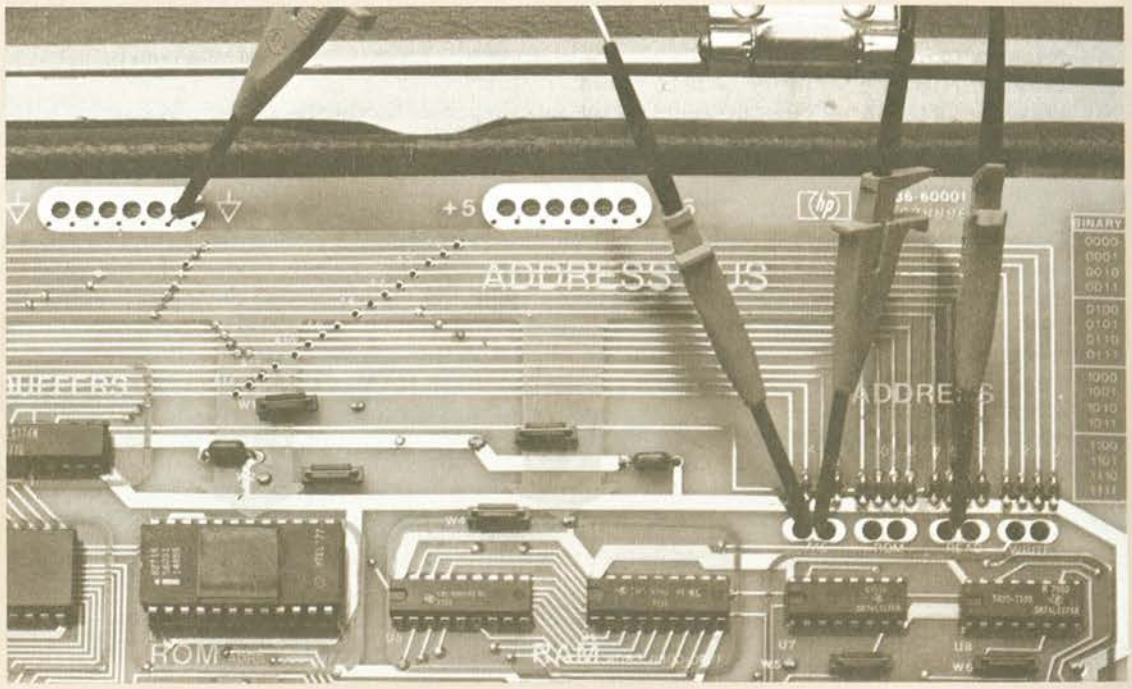


Figura 17-4. Come collegare al μLab l'analizzatore di firma nel caso del test degli indirizzi in free-run

tempo che intercorre tra il fronte di salita del bit A15 di indirizzo e il fronte di salita successivo dello stesso (l'intero campo di indirizzamento di 64K).

Il dato è acquisito dall'analizzatore di firma ad ogni fronte di salita della linea $\overline{\text{READ}}$, ossia 64K volte durante ciascuna finestra di misurazione (Figura 17-5).

C) L'analizzatore di firma può essere ora utilizzato per verificare che nel modo free-run gli indirizzi operino correttamente. Toccate la massa con la sonda dati (che agisce anche come sonda logica). Osservate che la firma è 0000. Questa è l'indicazione caratteristica dello stato logico 0, per qualunque modalità di collaudo.

D) Toccate la linea V_{cc} con la sonda dati in modo da rilevare la sua firma: osservate la firma 0001. Questa è la firma caratteristica dello stato logico 1, valida solo per questa configurazione dell'analizzatore di firma e del μLab. Potete verificare quanto appena detto, cambiando sull'analizzatore la posizione degli interruttori che indicano il fronte per START e STOP e osservando come varia la firma. Perciò l'aver visto 0001 indica che l'apparecchiatura era stata correttamente preparata e che il nucleo del μLab opera correttamente in free-run. Se questo non si fosse verificato, si poteva supporre o che la prova era stata effettuata in modo scorretto o che era presente un guasto nel nocciolo del μLab. Così facendo è facile circoscrivere i circuiti, presenti nel prodotto, che possono essere ritenuti possibili cause di malfunzione.

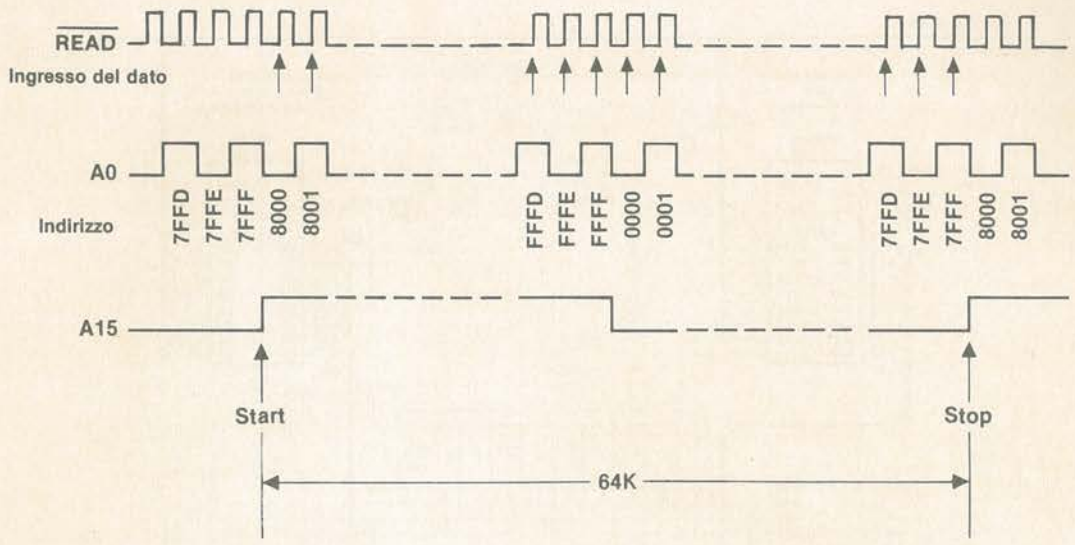


Figura 17-5. Nel corso del test degli indirizzi in free-run i dati vengono campionati entro finestre di misura di 64K indirizzi

E) Riferendovi alla Figura 17-6 e alla Tabella C-1, riportata in Appendice C, rilevate le firme sulle linee da A0 ad A15 del bus degli indirizzi e verificate che questi valori siano in accordo con quelli riportati nella tabella. Questo vi permette di stabilire che il microprocessore, il buffer (IC1) degli indirizzi, e il latch degli indirizzi (IC2) stanno operando in modo corretto.

F) Riferendovi alla Figura 17-7, leggete le firme dei piedini di uscita del decoder degli indirizzi (IC7 piedini 7,9,10,11,12,13,14,15) e confrontatele con i valori riportati in tabella. Queste firme confermano il corretto funzionamento del decoder degli indirizzi.

G) Tutti i valori riportati in Tabella C-1 vi danno informazioni utili. La ragione per cui, per un certo numero di dispositivi, non sono state riportate le firme di confronto per questo tipo di prova, sta nel fatto che tali dispositivi non vengono esercitati nel modo «free-run». O non c'è attività sulle loro uscite (devono ricevere un segnale di scrittura) o il dato, su di essi presente, non è prevedibile (come è il caso delle uscite della RAM). Perciò le firme di questi piedini non sono utili. Ricordate e verificate quanto detto quando eseguite i tre passi successivi.

H) Controllate con la sonda tutte le linee del bus dei dati e annotate le firme corrispondenti. Questa firma è il risultato del sommarsi di tutti i dispositivi del μLab, che accedono al bus dei dati durante la finestra dei 64K indirizzi.

I) Spegnete e riaccendete il μLab. Osservate che, probabilmente, le firme sulle diverse linee del bus dei dati sono cambiate. Questo si verifica perché, avendo spento il μLab, la RAM ora contiene nuovi dati casuali (dal momento che gli interruttori del bus dei dati sono aperti, il programma di accensione, che normalmente riempie la memoria con degli zeri, non può girare. Provate ancora. La firma probabilmente cambierà un'altra volta.

(continuazione)

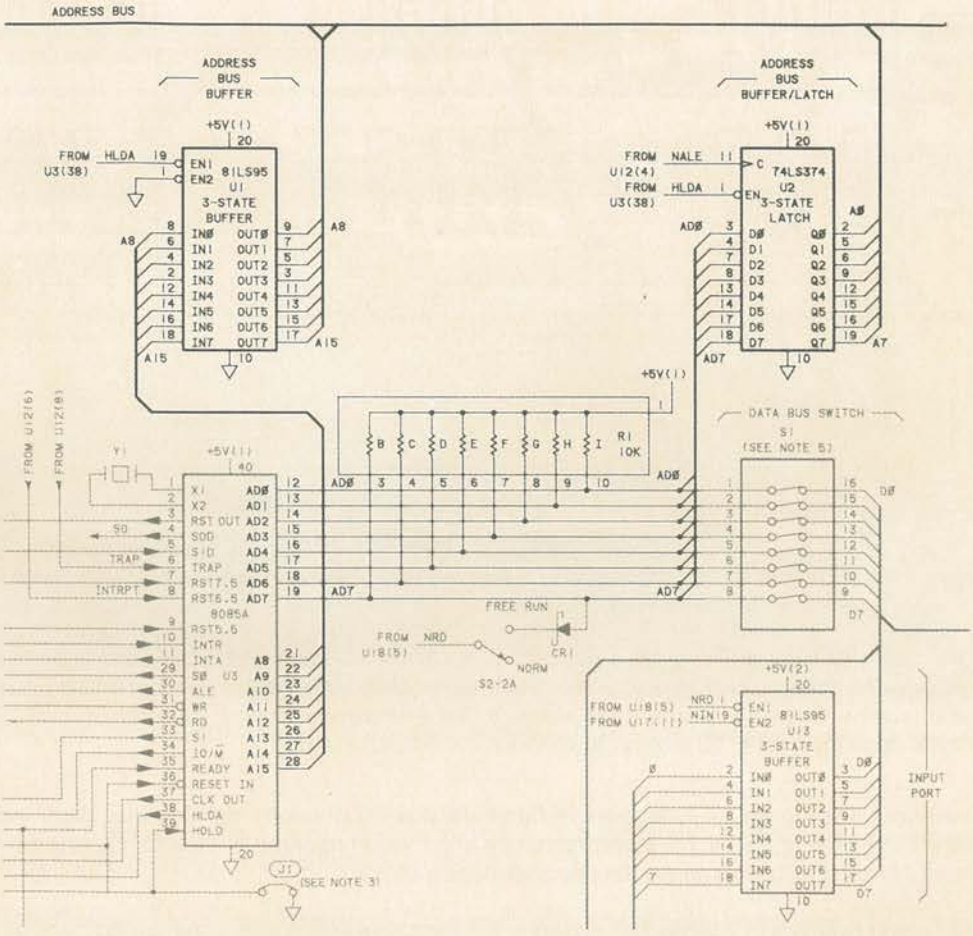


Figura 17-6. Circuito di pilotaggio e demultiplex degli indirizzi

J) Mantenendo la sonda sempre su una stessa linea del bus dei dati, commutate l'interruttore della porta di ingresso (non l'interruttore del bus dei dati) corrispondente a tale linea. L'interruttore d'ingresso si trova in basso a destra rispetto al display ed è collegato alla porta d'ingresso IC13. Osservate che la firma cambia in funzione della posizione di questo interruttore. Provate ancora su altre linee del bus: è importante notare che, su questa porta d'ingresso, può essere rilevata una certa attività del circuito sebbene non sia necessariamente l'attività corretta. Nel seguito, ulteriori esperimenti vi forniranno maggiori informazioni.

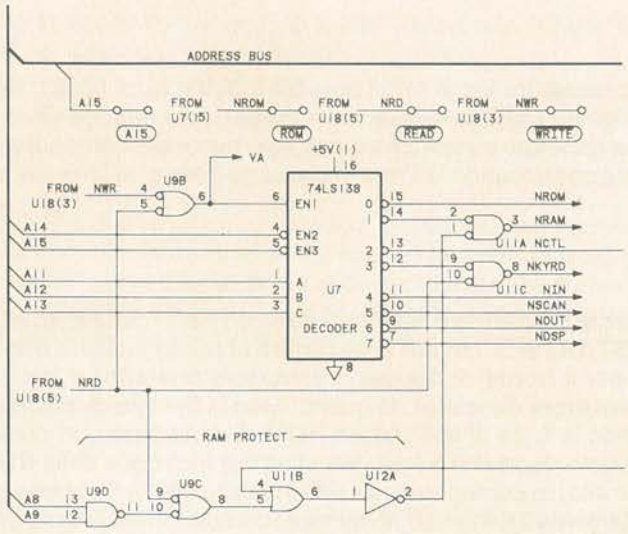


Figura 17-7. Circuito di decodifica degli indirizzi

RIASSUNTO

Avete fatto operare il μLab nel modo di prova «free-run»; il microprocessore è passato continuamente attraverso tutta l'area di memoria indirizzabile (64K). La preparazione dell'analizzatore di firma è stata verificata dalla lettura della firma di V_{cc}. I buffer e i decoder sono stati provati rilevando le firme sulle loro uscite.

Nel modo di prova free-run, è stato interrotto il percorso tra il bus dei dati e il microprocessore. Questo isolamento ha permesso che venissero letti tutti i dispositivi che colloquiano con il microprocessore (ROM, RAM, e porte d'ingresso) senza che tuttavia fossero influenzate le operazioni del microprocessore e mentre veniva incrementato il bus degli indirizzi. In questo modo, è stato possibile vedere l'effetto di alcuni di questi dispositivi, leggendo le firme del bus dei dati.

ESPERIMENTO 17-2

Prova della ROM durante il free-run

INTRODUZIONE

Nel modo free-run, il microprocessore legge tutti i possibili indirizzi (tutti i 64K). Se l'analizzatore di firma è predisposto in modo da analizzare i dati presenti sul bus dei dati solo quando viene indirizzata la ROM (indirizzi da 0000 a 07FF), le firme generate saranno funzione solamente del contenuto della ROM. LE 2K parole della ROM vengono verificate controllando la firma di ciascuna delle otto linee del bus dei dati.

PROCEDIMENTO

A) Verificate che il μLab si trovi ancora nel modo free-run (fate riferimento all'Esperimento 17-1). Collegate le sonde di START e STOP dell'analizzatore al punto indicato con $\overline{ROM}$. Predisponete l'ingresso di START per il fronte di discesa (interruttore premuto) e lasciate il pulsante STOP sul fronte di salita (interruttore rilasciato). In questo caso la finestra di misurazione inizia alla prima lettura in ROM (quando la linea di abilitazione ROM diventa bassa, in corrispondenza dell'indirizzo 0000) e si chiude giusto dopo che è stata letta l'ultima locazione della ROM (quando l'abilitazione della ROM ridiviene alta, in corrispondenza dell'indirizzo 0800). Sebbene questa finestra di misurazione contenga solamente 2K indirizzi, il microprocessore opera ancora in free-run, all'interno della lunga sequenza di 64K. Tuttavia l'analizzatore di firma sta ora prendendo in considerazione solamente quei 2K cicli della sequenza durante i quali la ROM è abilitata (Figura 17-8).

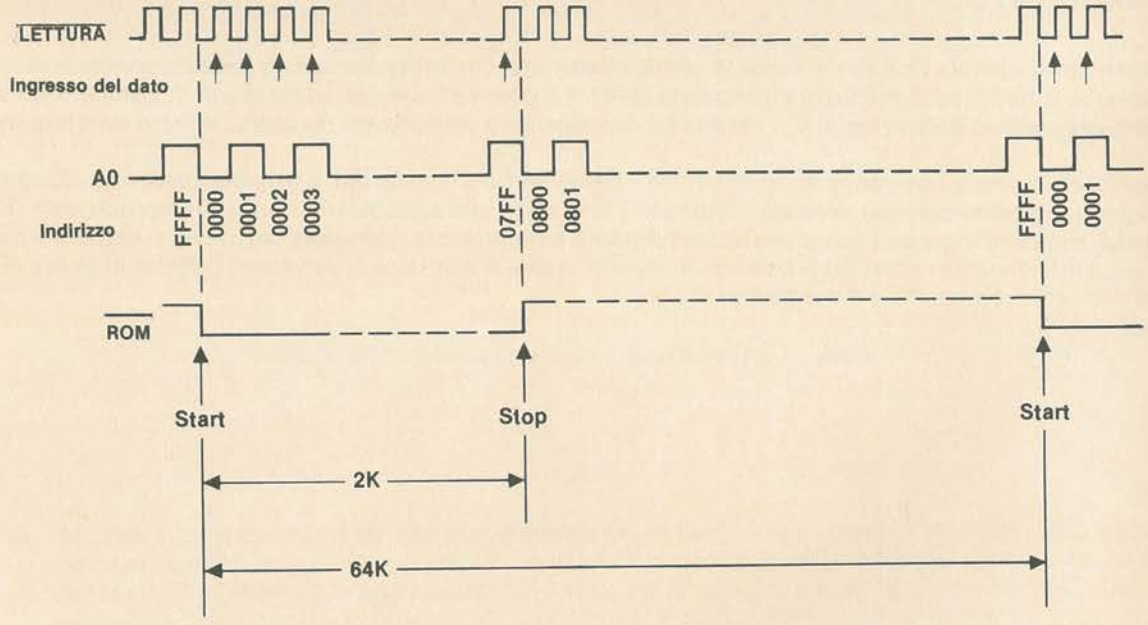


Figura 17-8. Nel corso del test della ROM in free-run i dati vengono campionati solo durante i 2K indirizzi in cui viene letta la ROM

B) Toccate V_{cc} con la sonda dati e verificate, riferendovi alla Tabella C-2, che sia presente la firma corretta (7A70) per il modo di prova free-run della ROM.

C) Verificate che le firme di ciascuna delle otto linee del bus dei dati concordino con quelle della tabella. Questa prova controlla tutti i 16K (2K × 8) bit della ROM.

D) Provate a modificare gli interruttori di ingresso, poi spegnete e riaccendete il μLab in modo da alterare i dati contenuti nella RAM. Le firme prese sul bus dei dati non cambiano. Gli altri dispositivi che comunicano con il bus dei dati sono ancora presenti, ma si trovano ora fuori dalla finestra di misurazione (definita da START e da STOP). Questa è una situazione diversa da quella descritta nel precedente esperimento nel quale la finestra di misurazione includeva tutti gli indirizzi e perciò tutti i dispositivi.

RIASSUNTO

Questa prova vi permette di ricavare numerose informazioni. Avete controllato la ROM e potete dire, con buone speranze, che è funzionante. È anche abbastanza sicuro che non si verificavano conflitti sul bus (due o più dispositivi che accedono al bus nello stesso istante) mentre veniva eseguito un accesso alla ROM (si sarebbe letta in tal caso una firma errata). Il bus dei dati non è bloccato né cortocircuitato e non presenta difetti seri (baffi di saldatura o di doratura, uscite di dispositivi bloccate, ecc.).

Tutte le informazioni diagnostiche osservate fino ad ora sono state ottenute facendo funzionare il sistema in modo free-run. Non sono state eseguite le istruzioni contenute in ROM (non è stato cioè eseguito alcun programma). Il μLab ha utilizzato solamente una piccola parte dei suoi circuiti (il «kernel» o nocciolo), operando ad anello aperto, in modo da fornire gli stimoli di prova (incrementando le linee di indirizzo e generando impulsi di lettura) a molti degli altri circuiti. Tali informazioni (nella forma di firme corrette) vi permettono di continuare con i prossimi collaudi, potendo già contare sulla corretta funzionalità di questi circuiti.

ESPERIMENTO 17-3

Loop di test per SA

Il loop di test per SA è un programma, inserito nella ROM del μLab, che ha il solo scopo di fornire uno stimolo controllato per l'analisi di firma. L'interruttore dell'analisi di firma (SA) è connesso ad una linea di interruzione del microprocessore. La routine di gestione di tale interruzione è costituita dal loop di test (anello di prova) per SA. L'interruttore SA pone quindi il μLab in questo modo di funzionamento; il tasto RESET ripristina invece il normale modo di lavoro. Nel modo test SA vengono anche accesi tutti i segmenti del display e i LED della porta d'uscita, che possono così essere direttamente controllati. Questo stato è anche un'indicazione implicita che nel μLab sta girando il programma di loop di test SA.

Questo loop genera dei segnali di stimolo per i dispositivi a cui il microprocessore accede con operazioni di scrittura (la porta di uscita, le porte di scansione e di display, l'altoparlante, e la logica di controllo). Nel loop di test vengono anche letti i dispositivi che possono inviare dati al microprocessore (gli interruttori della porta d'ingresso e la tastiera). Le risposte dei circuiti al programma di stimolo (costituito da una sequenza ripetitiva di semplici programmi, memorizzati in ROM, che esercitano i diversi dispositivi) sono rilevate dall'analizzatore di firma.

Un semplice programma ci sarà utile per illustrare il tipo di stimoli generati da questo loop di testa SA.
Di seguito è così riportata la sequenza di eventi eseguiti durante il test della porta d'uscita:

1. Accendi tutti i LED tranne D0
2. Accendi tutti i LED tranne D1

•
•
•

8. Accendi tutti i LED tranne D7
9. Accendi tutti i LED
10. Continua con il prossimo programma di stimolo, interno al loop di test SA.

Come potete vedere da questo esempio (che è tipico per molte routine di stimolazione), il programma esercita solo un dispositivo alla volta (in questo caso, la porta d'uscita IC15).

PROCEDIMENTO

I. Preparazione del μLab per il loop di test SA

A) Spostate verso il basso, nella sua posizione di riposo, l'interruttore di free-run (FR), in modo da liberare il microprocessore dall'istruzione che forzava il free-run.

B) Muovete verso il basso tutti e otto gli interruttori del bus dei dati, in modo da ripristinare il collegamento tra il bus dei dati del microprocessore e il resto del sistema (anello-chiuso). Ora il μLab è stato riportato nella sua configurazione normale di lavoro.

C) Premete due volte RESET. Sul display appare uLAbUP. Il sistema ora opera nel suo modo normale. Essendosi prima aperto e poi ricollegato il bus dei dati, i LED di uscita si trovano in uno stato logico non prevedibile.

D) Muovete verso l'alto e poi ancora verso il basso l'interruttore SA (posto alla sinistra dell'interruttore FR). Prima di fare questo controllate che sul display sia visualizzato uLAbUP, per assicurarvi che la linea d'interruzione, collegata all'interruttore SA, sia abilitata dal microprocessore. Quando muovete verso l'alto l'interruttore SA udirete un suono di avvertimento (bip). Osservate che i segmenti del display ed i LED d'uscita sono tutti accesi e che si sente un debole scricchiolio provenire dall'altoparlante. Ora il μLab si trova nel loop di test SA.

II. Lettura delle firme nel test SA di scrittura

A) Toccate, con il puntale dell'analizzatore di firma, una qualsiasi linea di segnale dei LED di uscita (Figura 17-9). Il lampeggiare del puntale rivela che sono presenti degli impulsi, anche se i LED sembrano completamente accesi. Questo comportamento si spiega con il fatto che la routine di prova SA tiene accesi i LED per la maggior parte del tempo, mentre li spegne solo di tanto in tanto permettendo così di controllare, con l'analizzatore di firma, tutti e due gli stati logici d'uscita di IC15.

Figura 17-9. Verifica dei LED d'uscita durante il loop di test SA

B) Collegate entrambi i cavi di START e di STOP, dell'analizzatore di firma, al foro di prova A15. Connettete il cavo di CLOCK al foro $\overline{\text{WRITE}}$. Assicuratevi che il cavo GND sia ancora connesso alla massa.

C) Predisponete sull'analizzatore di firma gli ingressi di START, STOP e CLOCK per il fronte di salita (interruttore rilasciato). Il bit A15 è controllato dal programma loop di test SA e su di esso è presente un impulso una sola volta durante ciascun ciclo del loop. Il programma genera questo impulso effettuando un'operazione di scrittura all'indirizzo 8000 all'inizio di ogni loop di test. L'analizzatore di firma viene sincronizzato dalla linea $\overline{\text{WRITE}}$ del μLab e perciò acquisisce i dati ogni volta che si verifica un'operazione di scrittura.

(continuazione)

D) Controllate la firma di V_{cc} (fate riferimento alla Tabella C-3) e verificate così che sia tutto pronto per il test SA di scrittura.

E) Utilizzando la Tabella C-3, controllate le firme di ogni nodo che ritenete significativo. Notate che, in questo modo di test, i dispositivi d'ingresso (che sono abilitati dal segnale $\overline{READ}$) non vengono verificati.

F) Mentre state controllando le varie linee del bus dei dati, muovete a caso gli interruttori della porta d'ingresso e premete uno qualsiasi dei tasti della tastiera (tranne RESET). Verificate come queste operazioni non abbiano alcun effetto sulle firme. Questi interruttori non influiscono perché sono abilitati dal segnale $\overline{READ}$, mentre l'analizzatore di firma è sincronizzato con il segnale di $\overline{WRITE}$.

III. Il test SA di lettura

Il loop di test SA genera due tipi di stimoli. Nella parte precedente di questo esperimento, il loop di test veniva usato per scrivere dei dati dal microprocessore ai dispositivi I/O e alla memoria. Il loop di test SA può anche far sì che il microprocessore legga dei dati da questi dispositivi. Il punto, cui è collegato l'ingresso del CLOCK dell'analizzatore di firma, determina se le firme prelevate dipendono dal dato letto o dal dato scritto. I fori di prova $\overline{READ}$ e $\overline{WRITE}$ vengono utilizzati a questo scopo.

A) Spostate il cavo del CLOCK, dell'analizzatore di firma, dal punto $\overline{WRITE}$ a quello $\overline{READ}$ del μLab. Verificate che gli ingressi START, STOP e CLOCK sull'analizzatore di firma siano ancora predisposti per il fronte di salita e che i cavi di START e di STOP siano ancora collegati al punto di prova A15. Ora potete rilevare le firme dei diversi dispositivi letti dal microprocessore.

B) Toccate V_{cc} con il puntale e controllate la firma facendo riferimento alla Tabella C-4. Avete predisposto lo strumento in un modo che vi potrebbe sembrare familiare perché si hanno gli stessi collegamenti e le stesse disposizioni dei fronti che si avevano nel modo free-run. Le firme di V_{cc} sono però diverse, in questo modo di operazione, poiché esiste una differenza per quanto riguarda la lunghezza del ciclo del programma. Nel caso del modo free-run, A15 definiva un ciclo di lunghezza pari a 64K indirizzi successivi. Nel caso del loop di test SA, il programma di stimolazione SA si serve della linea A15 per controllare START e STOP. In questo caso la lunghezza della finestra è molto minore di 64K cicli perché l'anello di prova è molto più breve.

C) Spostate verso il basso tutti gli interruttori di ingresso, in modo tale che siano presenti degli 0 logici su tutte le linee del bus dei dati quando viene letta la porta d'ingresso. Verificate che le firme, sulle linee del bus dei dati, coincidano con i valori della Tabella C-4.

D) Riportate nella posizione iniziale tutti gli interruttori d'ingresso. La firma su ciascuna linea del bus dei dati cambierà in corrispondenza del cambiamento dell'interruttore d'ingresso. Queste firme devono concordare con quelle riportate nella tabella relativa a questa nuova predisposizione; così facendo controllate che gli interruttori d'ingresso vengano letti correttamente sul bus dati.

E) Riferitevi alla Figura 17-10 e alla Tabella C-4. Verificate che i tasti riportati in tabella (tranne il tasto RESET) funzionino correttamente ed osservate le firme corrispondenti sugli ingressi dei dati di IC18 e sulle linee del bus dei dati.

F) Premete INTRPT e verificate che l'altoparlante emetta dei suoni (bip) ripetuti. Il programma di stimolazione SA vi dà anche questa verifica «acustica».

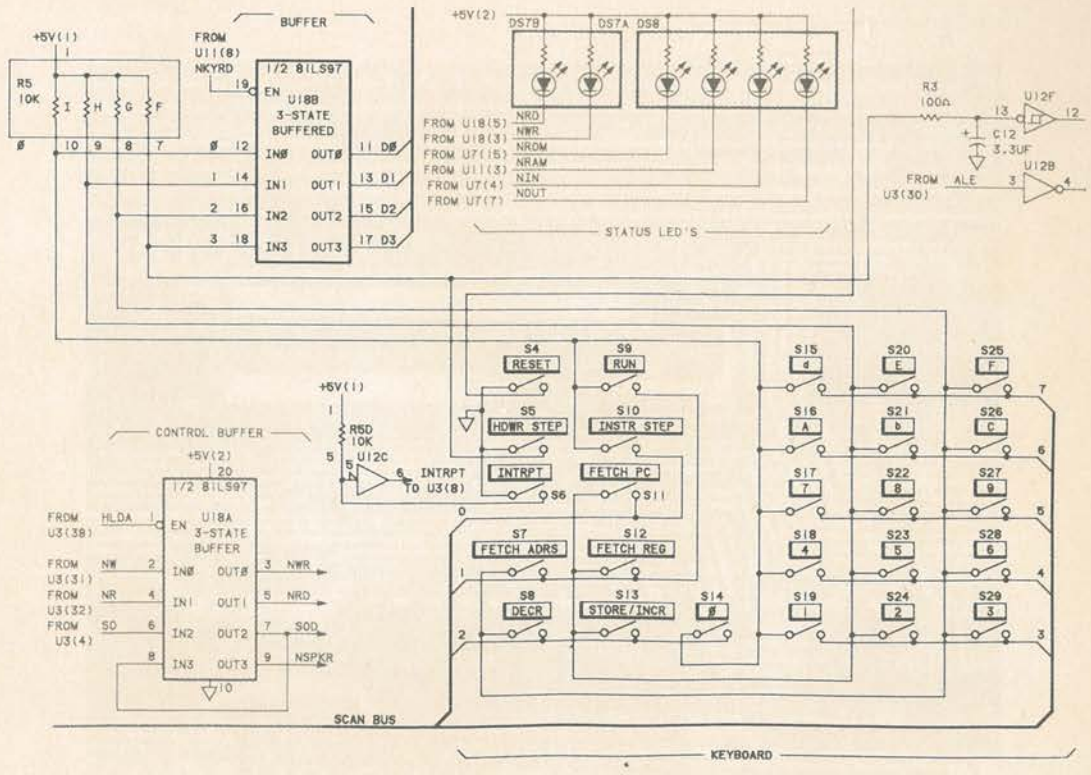


Figura 17-10. Circuito di ingresso tastiera

IV. Localizzazione di un semplice guasto

A) Localizzate il ponticello 10 di simulazione guasto (indicato con W10) posto vicino al centro della scheda del μLab (Figura 17-11). Rimuovetelo ed osservate che il LED 4 di uscita si spegne.

B) Riferendovi alla Figura 17-12 e alla Tabella C-3 delle firme, controllate la firma sul LED 4 (assicuratevi di cambiare la disposizione della prova secondo quanto specificato in Tabella C-3). La firma errata che potete vedere in questo punto significa che il LED probabilmente non è guasto, ma che non sta ricevendo un segnale corretto.

C) Riferendovi allo schema, seguite a ritroso il percorso del segnale, fino ad arrivare alla porta di uscita IC15-12. Osservate che anche questo piedino di uscita ha ancora la stessa firma scorretta. Non avete ancora trovato il guasto.

ESPERIMENTO 17-3

(continuazione)

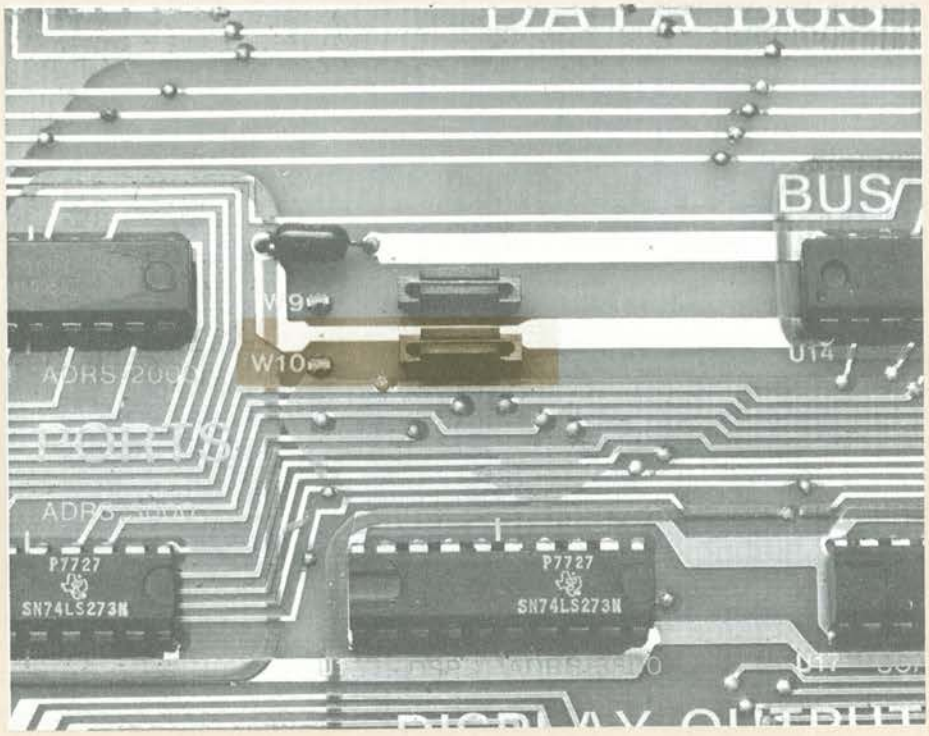


Figura 17-11. Posizione del ponticello W10 di guasto

D) Esaminate l'ingresso D4 (IC15-13) della porta di uscita. Anche questa firma non è corretta.

E) Ora, sempre per la linea D4, verificate l'uscita del buffer del bus (IC14-11). Questa firma è buona. Perciò il guasto deve trovarsi tra questa uscita e IC15-3. Avete così circoscritto ad una sola pista del circuito stampato l'area in cui si può trovare il guasto.

F) Analizzate il segnale lungo la pista che va da IC15-13 verso IC14-11. Ora avete trovato il guasto.

G) Ripristinate, tra i due fori di destra corrispondenti alla locazione del guasto W10, il ponticello 10 di simulazione guasto.

(continuazione)

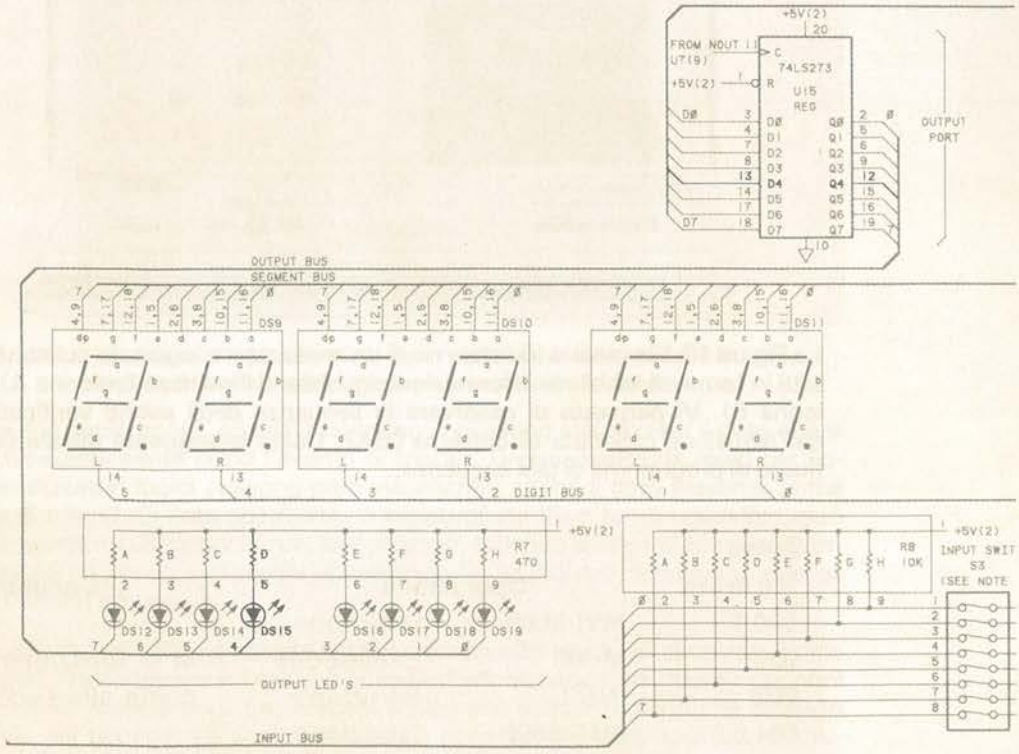


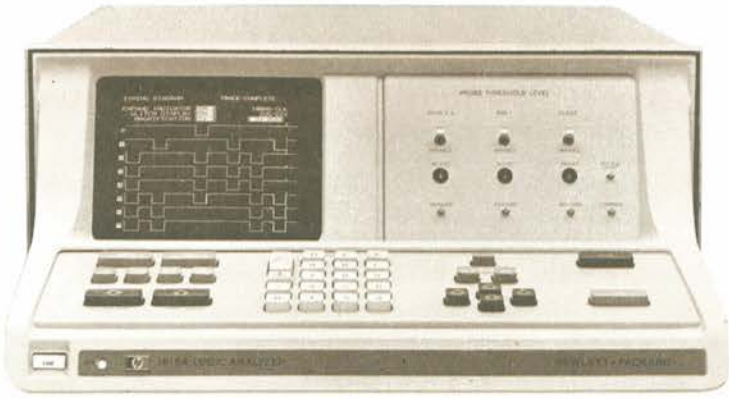
Figura 17-12. Circuito della porta d'uscita

RIASSUNTO

In questo esperimento avete imparato ad usare l'analizzatore di firma per verificare il comportamento di un sistema a microprocessore. Sono state discusse alcune tecniche di ricerca guasti tra cui il modo di test free-run ed un programma speciale loop di test SA. Mediante queste tecniche, per poter provare altri circuiti chiave del sistema (incluse le ROM), deve operare correttamente solo una piccola parte di tutto il circuito (il nucleo). Una volta che il nucleo è stato verificato, si è utilizzato il programma loop di test SA per controllare i dispositivi non collaudati in free-run. Avete visto come l'analizzatore di firma, insieme ad una comprensione globale del funzionamento dei dispositivi del μLab (e ad un certo buon senso nell'eseguire la ricerca dei guasti), vi permetta di ritrovare un nodo guasto lavorando solo un poco sull'hardware e non toccando neppure il software.

ANALIZZATORI LOGICI

Gli analizzatori logici (Logic Analyzer) sono strumenti specializzati usati per esaminare dei gruppi di segnali logici (generalmente 16 o 32) che si verificano nelle lunghe e complesse sequenze di dati proprie dei sistemi a microprocessore. Questo paragrafo spiega, in termini generali, che cosa siano gli analizzatori logici e come essi vengano utilizzati. Sebbene siano usati principalmente per lo sviluppo di sistemi a microprocessore e per ricerche guasti a livello sistema, si possono anche avere applicazioni in situazioni di assistenza sui prodotti.



La Figura 17-13a mostra lo schermo di un analizzatore logico, in cui sono rappresentate in forma di tabella le informazioni riguardanti l'indirizzo (colonna A) e i dati (colonna B). Vi permette di osservare la sequenza degli eventi verificatisi a partire dall'istruzione collocata all'indirizzo 06A2. Usate la seguente tabella per seguire il flusso di programma di tale sequenza:

Linea Numero	Operazione		Commenti
000,1	MVI M,00	Istruzione	
002	M←00	Esecuzione	M = locazione 0BF0
003	INR L	Istruzione	punta alla successiva M
004,5,6	JMP 069B	Istruzione	
007	INR M	Istruzione	
008	μP←M	Esecuzione	M = locazione 0BF1
009	M←M + 1	Esecuzione	
010	MOV A,M	Istruzione	
011	A←M	Esecuzione	M = locazione 0BF1
012,13	CP1 0A	Istruzione	
014	JNZ	Istruzione	

Queste sedici linee, visualizzate sul display, rappresentano solo una piccola sequenza all'interno di un lungo programma. L'analizzatore logico «cattura» questa sequenza, la memorizza nella sua memoria interna e la può così visualizzare indefinitamente. L'analizzatore logico vi permette anche di osservare l'attività che si verifica prima o dopo le operazioni mostrate.

Un analizzatore logico è simile sotto molti aspetti ad un oscilloscopio; possiede dei segnali d'ingresso, un trigger (sincronismo), circuiterie di temporizzazione, ed uno schermo (tubo a raggi catodici, Cathode Ray Tube, CRT). I segnali d'ingresso differiscono da quelli di un oscilloscopio perché nel caso dell'analizzatore logico sono presenti da 16 a 32 ingressi, sensibili a soglie logiche, che rivelano i livelli logici 1 o 0.

Il circuito di trigger è molto più sofisticato di quello di un oscilloscopio. Gli analizzatori possono sincronizzarsi su di un certo indirizzo direttamente, dopo che questo si

è presentato N volte (nel caso di loop software), dopo che sono passati N cicli di clock a partire da quell'indirizzo (nel caso di routine di ritardo) o dopo una certa sequenza di indirizzi di trigger (per rivelare un particolare percorso del programma).

Le parole di trigger non devono essere necessariamente degli indirizzi. Possono essere dati, linee di controllo o qualsiasi insieme di segnali logici.

LINE NO.	A HEX	B HEX
001	06A3	00
002	0BF0	00
003	06A4	2C
004	06A5	C3
005	06A6	9B
006	06A7	06
007	069B	34
008	0BF1	03
009	0BF1	04
010	069C	7E
011	0BF1	04
012	069D	FE
013	069E	0A
014	069F	C2

Figura 17-13a. L'analizzatore logico è utilizzato per visualizzare indirizzi e dati in forma tabellare ed è così possibile seguire l'esecuzione del programma

Similmente agli oscilloscopi, gli analizzatori logici possono visualizzare il dato che si presenta immediatamente dopo l'evento di trigger. Diversamente da molti oscilloscopi, gli analizzatori logici possono però visualizzare anche il dato presente prima dell'evento di trigger. Questa possibilità, la registrazione di un *tempo negativo*, può essere positivamente usata per la ricerca dei guasti, facendo sì che un'operazione errata del sistema sia l'evento di trigger ed osservando, all'indietro, quali siano stati gli eventi che l'hanno provocata.

Per le temporizzazioni, gli analizzatori logici si servono del clock del circuito al quale sono collegati. Viene generata una nuova linea d'informazione sul display per ogni impulso di clock. Mentre degli oscilloscopi si dice che sono strumenti che operano nel «dominio del tempo», gli analizzatori logici sono spesso qualificati come strumenti che operano nel «dominio dei dati».

In un oscilloscopio, il tempo scorre attraverso lo schermo, da sinistra verso destra, utilizzando una base dei tempi interna. In un analizzatore logico, il tempo scorre dall'alto verso il basso, una linea per ogni ingresso del clock. Un analizzatore visualizza un elenco di dati digitali. In Figura 17-13a si ha una presentazione in esadecima-

LINE NO.	A BIN	B BIN
001	0000011010100011	00000000
002	0000101111110000	00000000
003	0000011010100100	00101100
004	0000011010100101	11000011
005	0000011010100110	10011011
006	0000011010100111	00000110
007	0000011010011011	00110100
008	0000101111110001	00000011
009	0000101111110001	00000100
010	0000011010011100	01111110
011	0000101111110001	00000100
012	0000011010011101	11111110
013	0000011010011110	00001010
014	0000011010011111	11000010

Figura 17-13b. Le stesse informazioni della Figura 17-13a ma presentate in forma binaria

le, ma si può anche avere una visualizzazione in binario, ottale o decimale. La Figura 17-13b mostra, rappresentata in binario, la sequenza del programma della 17-13a. Non vengono visualizzati né i valori né le forme d'onda delle tensioni analogiche.

Alcuni analizzatori logici hanno una caratteristica chiamata *timing analysis* (analisi temporale). Con l'ausilio di una base interna dei tempi, possono essere visualizzati sullo schermo, in modo simile ad un oscilloscopio (Figura 17-14) i diagrammi dei tempi dei segnali logici di ingresso. La differenza consiste nel fatto che la rappresentazione è «digitalizzata» (ovvero approssimata) a valori di tensione e tempo discreti onde fornire una chiara visualizzazione grafica, senza dare invece informazioni specifiche sulla forma d'onda. Possono essere anche rilevati gli spifferi (Glitch, impulsi molto brevi) per mezzo di un rilevatore interno e di circuiti di allungamento. Si possono osservare molti ingressi per volta. Utilizzando completamente la possibilità del trigger, potete osservare la temporizzazione di una sezione limitata all'interno di una lunga sequenza di programma. Il trigger può operare in modo così sofisticato grazie al fatto che le forme d'onda sono memorizzate in forma digitale nella RAM interna.

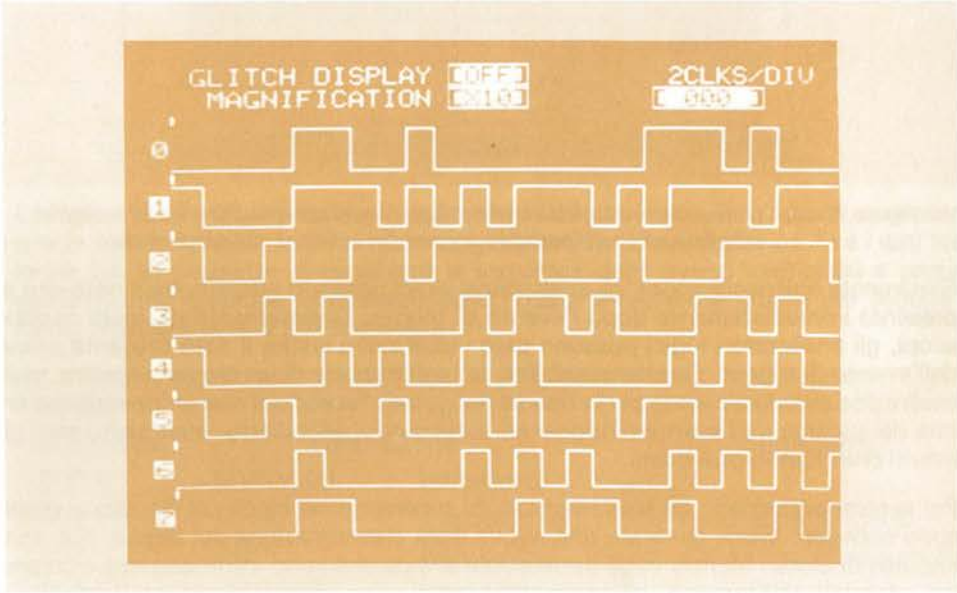


Figura 17-14. Le otto linee del bus dei dati visualizzate nel modo «analisi temporale»

Un'altra caratteristica di molti analizzatori logici è il modo mappa (Map Mode), utile per la manutenzione di un prodotto. In questo modo, il video viene usato come uno schemo X-Y costituito da 256 × 256 punti (65.536 o 2^{16}). Ciascun punto è usato per rappresentare un indirizzo nello spazio di 64K della memoria. Quando un sistema a microprocessore sta eseguendo un programma, ogni volta che il microprocessore accede ad un indirizzo viene acceso uno dei 64K punti del video. Quanto più spesso il microprocessore fa un riferimento ad un certo indirizzo, tanto più luminoso diventa il punto corrispondente. Quando un programma esegue delle istruzioni in sequenza, sullo schermo vengono illuminati dei punti contigui. Quando si verificano dei salti nel programma, questi possono essere invece osservati sul video come un salto da un insieme di punti ad un altro. Per migliorare la risoluzione, il display può essere espanso in modo da ingrandire la regione d'interesse. La Figura 17-15 mostra un loop di programma del'8085 visualizzato ingrandito. Il programma inizia all'indirizzo 0800. Quando arriva all'indirizzo 0808, memorizza un byte di dati all'indirizzo 3000. All'indirizzo 0810 salta a 0F20 per continuare il programma. L'indirizzo 0F20 contiene un'istruzione che provoca la memorizzazione di un byte di dati all'indirizzo 3838. Infine, una istruzione di salto in 0F24 fa tornare indietro il programma all'indirizzo 0800 per ripetere la sequenza.

La visualizzazione della mappa mostra i punti in cui il sistema si trova (cioè quali subroutine e indirizzi di periferiche sta usando). Rispetto a questo tipo base di modo mappa, ci sono molte variazioni e miglioramenti che possono essere presenti in particolari analizzatori logici.

Confrontate la Figura 17-15 con la Figura 17-16 (free-run). Nel modo free-run, il microprocessore si trova un uguale tempo in tutti gli indirizzi e di conseguenza sul video vengono accesi tutti i punti-indirizzo. La Figura 17-15 mostra un flusso di programma evidente, mentre la Figura 17-16 non ne mostra nessuno. Le mappe sono utili per controllare un sistema a microprocessore che mostri una qualche attività sul bus ma che non stia tuttavia funzionando correttamente. La mappa di un prodotto funzionante può anche essere confrontata con quella di uno non funzionante, in modo da essere di ulteriore aiuto ad identificare le possibili aree del problema.

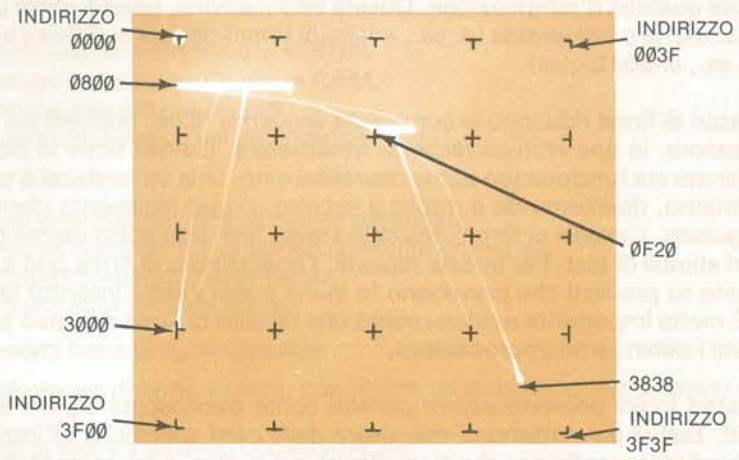


Figura 17-15. L'analizzatore logico viene utilizzato nel modo «a mappa espansa» al fine di mostrare l'attività degli indirizzi

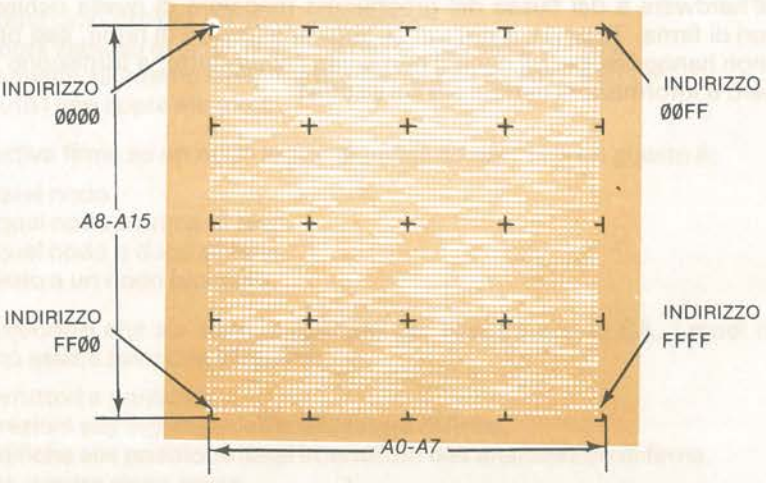


Figura 17-16. La mappa visualizzata durante il modo free-run indica una uguale attività per tutti i 64K indirizzi.

Gli analizzatori logici sono utili per lo sviluppo del software di un sistema, grazie alle loro capacità di sincronizzarsi e di seguire programmi specifici e sequenze di tempo, su un prototipo che stia operando. Sempre tali analizzatori possono generare una mappa che indica i punti in cui il sistema a microprocessore si sta trovando. Queste stesse caratteristiche possono essere usate anche per analizzare prodotti già esistenti. Possono così essere osservate sul prodotto in collaudo delle parti di programmi, posto che l'apparecchiatura sia stata correttamente preparata. Viene generalmente richiesta una buona documentazione d'assistenza che descriva come si deve preparare lo strumento, le uscite visualizzate, e le azioni da intraprendere quando viene notata un'attività errata nel flusso del programma.

Lezione 17

Il flusso costante di segnali sopra i bus di un sistema a microprocessore dà luogo ad una notevole quantità d'informazione. Questa informazione, perché abbia un senso, deve essere ridotta o compressa (p. es., analisi di firma) oppure suddivisa in parti più piccole (p. es., analisi logica).

Gli analizzatori di firma riducono le complesse sequenze di bit, presenti nei sistemi a microprocessore, in una «firma» facile da interpretare. Queste firme vi dicono se il nodo esaminato sta funzionando come dovrebbe o no. Una volta che si è trovato un nodo non buono, ripercorrendo a ritroso il circuito, si può facilmente identificare la causa del guasto. L'analisi di firma utilizza lo stesso prodotto sotto esame per generare i propri stimoli di test. Per questa ragione, l'analizzatore di firma può meglio essere utilizzato su prodotti che prevedano (o in cui possa essere inserito) tale tipo di collaudo. È molto importante rendersi conto che l'analisi di firma non può essere applicata a tutti i sistemi a microprocessore.

Gli analizzatori logici possono essere pensati come oscilloscopi digitali altamente specializzati. Essi vi permettono di esaminare delle parti specifiche all'interno delle sequenze hardware e software che si verificano nei sistemi a microprocessore. Possono essere utilizzati in particolare come strumenti di supporto per lo sviluppo di nuovi prodotti. La caratteristica del modo mappa è invece quella di essere rivolti alla ricerca guasti in prodotti già esistenti. Gli analizzatori logici richiedono una comprensione dell'hardware e del flusso del programma maggiore di quella richiesta dagli analizzatori di firma. Tuttavia, a differenza degli analizzatori di firma, essi per essere utilizzati non hanno bisogno di circuiti particolari nel prodotto, e forniscono un maggior numero d'informazioni dettagliate sul sistema.

Lezione 17

1. Nel modo di prova free-run, le configurazioni di stimolo del circuito sono generate da:
   a. un programma memorizzato in ROM.
   b. un programma memorizzato in RAM.
   c. un'azione casuale dei circuiti.
   d. una sequenza ciclica del microprocessore.
2. Le firme possono:
   a. fornire informazioni circa la natura del guasto.
   b. dire se un nodo sta funzionando correttamente o no.
   c. verificare i preparativi della prova.
   d. fare tutto quanto detto sopra.
3. Una corretta firma di V_{cc} vi dice che:
   a. l'analizzatore di firma è stato, con buona probabilità, correttamente predisposto.
   b. il loop di test sta girando correttamente.
   c. c'è un livello logico «1» sulla linea V_{cc}.
   d. sono vere tutte le affermazioni precedenti.
4. Un dato viene elaborato da un analizzatore di firma:
   a. ad ogni ciclo di clock del microprocessore.
   b. ad ogni impulso di clock che entri nell'analizzatore di firma.
   c. solamente all'interno della finestra limitata da START e STOP.
   d. in tutti i casi sopra elencati.
5. Una cattiva firma su un nodo logico significa sempre che un guasto è:
   a. su quel nodo.
   b. su quel nodo o prima di esso.
   c. su quel nodo o dopo di esso.
   d. dovuto a un nodo bloccato.
6. In un prodotto che sia stato progettato per operare con la SA, i modi di prova possono essere selezionati tramite:
   a. interruttori e ponticelli.
   b. correzioni sull'ingresso dell'analizzatore di firma.
   c. modifiche alla posizione degli interruttori dell'analizzatore di firma.
   d. tutto quanto detto sopra.
7. La ragione per cui un analizzatore logico può catturare e visualizzare delle sequenze particolari di un programma, meglio di un oscilloscopio, sta nel fatto che gli analizzatori logici hanno una memoria interna e:
   a. circuiti di sincronismo sequenziali su molte linee.
   b. un clock interno veloce.
   c. un video ad alta risoluzione.
   d. tutto quanto detto prima.
8. Il modo mappa vi permette di vedere:
   a. la locazione dei componenti fisici.
   b. in quali indirizzi il sistema sta operando.
   c. quale dato è posto ad un certo indirizzo.
   d. un listing parziale del programma.

LEZIONE 18

Ricerca di guasti nei sistemi a microprocessore

INTRODUZIONE

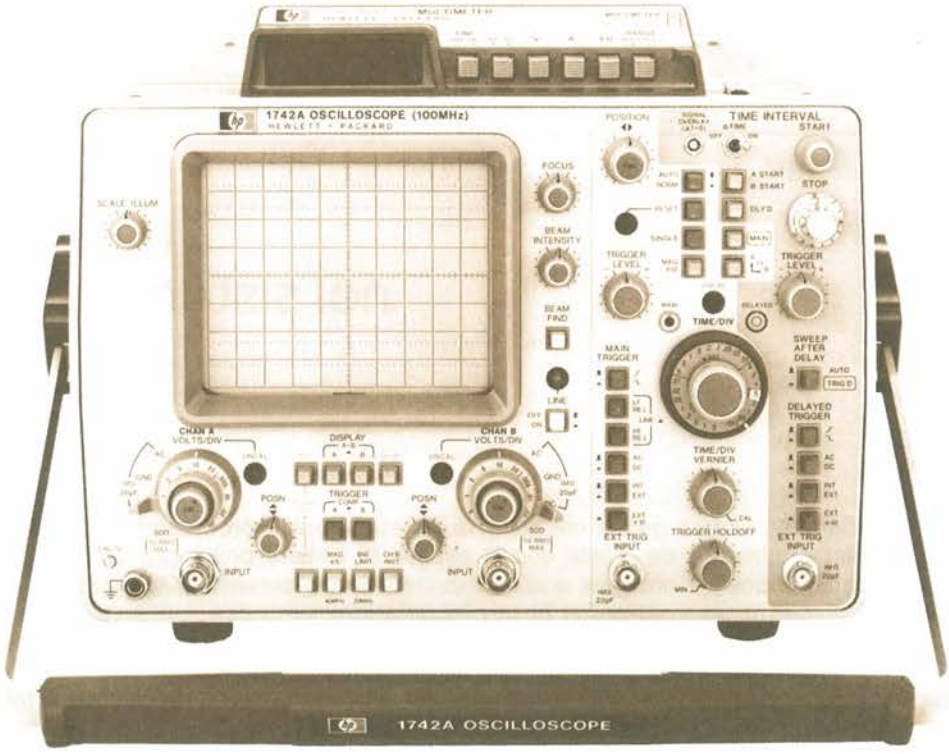
La filosofia della ricerca guasti nei prodotti basati sui microprocessori non differisce fondamentalmente da quella che si applica ai sistemi progettati con circuiti digitali convenzionali. Come per qualsiasi circuito in cui cercate di analizzare o ricercare i guasti, è utile per prima cosa prendere familiarità con il circuito stesso. Studiando il funzionamento teorico, il diagramma a blocchi e lo schema circuitale si ottiene una buona conoscenza di base da cui è possibile partire per lavorare. In questa lezione vengono discussi i problemi che si possono avere nei sistemi a microprocessore e le tecniche di ricerca guasti utilizzabili.

PROBLEMI DI RICERCA GUASTI NEI MICROPROCESSORI

Esistono alcuni problemi di collaudo che sono pressoché peculiari dei sistemi a microprocessore. Per prima cosa, la parte principale del controllo è realizzata attraverso il software, per cui è difficile seguire il flusso dei segnali. Un'altra difficoltà è legata al fatto che ogni evento si manifesta troppo rapidamente per poter essere seguito in tempo reale. Nella gran parte dei casi, un sistema a microprocessore, diversamente da molti dei circuiti logici, non può essere fermato o manipolato. Le misure devono essere effettuate mentre il microprocessore sta funzionando. Questa esigenza particolare limita l'utilità della sonda logica e del generatore d'impulsi, ma evidenzia l'utilità del rivelatore di corrente, dell'oscilloscopio, dell'analizzatore di firma e dell'analizzatore logico perché questi strumenti, per eseguire le misure, richiedono che i circuiti siano attivi.

Ulteriori difficoltà sono poste dalle strutture a bus dei microprocessori. Spesso il dato su questi bus è instabile o privo di significato a causa delle uscite a tre-stati, del multiplexing e dei transitori causati dalle commutazioni. Queste condizioni non provocano problemi al sistema in se stesso, dal momento che questo è sincrono e sa quando le linee dei bus portano dei segnali stabili. Anche l'analizzatore di firma e l'analizzatore logico sanno, in funzione del segnale di clock ad essi fornito, quando queste linee sono valide. L'oscilloscopio non ha invece questa capacità. Esso può dar ben poche informazioni quantitative, ma è utile per esaminare dei fattori qualitativi riguardanti, ad esempio, l'attività generale, i livelli logici, le forme d'onda temporali e i conflitti sul bus.

Dal momento che le strutture a bus fanno sì che molti dispositivi siano collegati ad un singolo nodo, può essere difficile trovare un dispositivo guasto su un nodo di tale tipo. In questi casi si mostra utile il rivelatore di corrente. Il bus dei dati agisce anche come percorso di reazione dei segnali digitali e tende a propagare gli errori attraverso circuiti correttamente funzionanti, riportandoli poi al circuito responsabile della malfunzione.



L'oscilloscopio è uno strumento che aiuta ad identificare dei problemi nei sistemi a microprocessore.

Il modo migliore per trattare questo problema è quello di aprire, quando possibile, il percorso di reazione. In questa lezione vengono discusse le tecniche che permettono di realizzare queste operazioni.

Spesso sono dei dispositivi complessi ad essere connessi ai bus del microprocessore. Utilizzando dei semplici test stimolo-risposta, è difficile controllare questi dispositivi. Il loro funzionamento corretto può essere verificato sostituendoli con un integrato già riconosciuto come funzionante e verificando che la funzione che essi realizzano nel sistema venga in tal caso eseguita correttamente.

I microprocessori sono macchine sequenziali. Il flusso del programma dipende da una lunga sequenza di istruzioni ed eventi. L'intero sistema può non funzionare correttamente anche se è sbagliato un solo bit di informazione. Nel caso di errori in un solo bit, le cause più comuni sono costituite dagli impulsi spuri e dal verificarsi di bit non funzionanti nella memoria. In questa lezione vengono discussi altri tipi di errore. Questi guasti sono difficili da evidenziare perché può sembrare che sia l'intero sistema a lavorare nel modo sbagliato.

L'esperienza, ricavata realizzando gli esperimenti di ricerca guasti sul μLab presentati nella Lezione 19, vi dà una buona base per imparare ad eseguire la manutenzione di altri prodotti basati sui microprocessori. Questa esperienza può evitare che i problemi veramente difficili della ricerca guasti siano da voi messi da parte (o anche peggio). Le cose nuove sembrano sempre molto difficili al primo istante e anche per i microprocessori si verifica la stessa cosa. Strumenti appositamente progettati ed una buona documentazione fornita dal costruttore possono rendere molto più facile la ricerca dei guasti. L'uso dell'analisi di firma e di altri sofisticati strumenti di supporto può ridurre notevolmente il lavoro della ricerca guasti.

Esistono dozzine di microprocessori diversi e centinaia di persone progettano prodotti e relative procedure di manutenzione. Dal momento che il μLab è progettato specificamente per scopi educativi e per l'insegnamento della ricerca guasti, i concetti sviluppati utilizzando il μLab dovrebbero essere applicabili a molte classi di sistemi a microprocessore. Esso è molto simile a un tipico (ma piccolo) sistema, ed è altrettanto reale.

PROBLEMI SPECIFICI DEI SISTEMI A MICROPROCESSORE

Clock

Dei cattivi segnali di clock possono far sì che il sistema «funzioni» ma solo in modo folle. Esistono un certo numero di motivi per cui possono nascere dei problemi nel segnale di clock del sistema. Problemi relativi al clock possono far sì che il sistema non funzioni per nulla (nessuna attività), che funzioni solo ad anello aperto (free-run), o che manifesti una attività parziale (una sequenza di programma indefinita e priva di significato). Alcuni microprocessori sono sensibili alla frequenza del clock. Dal momento che molti sistemi girano al limite delle specifiche, una variazione seppur piccola nella frequenza del clock (troppo veloce) può causare guasti al sistema. Se il sistema funziona troppo lentamente, possono invece non operare correttamente le celle delle memorie dinamiche del sistema stesso. Entrambi questi problemi si verificano frequentemente quando vengono usati dei circuiti di clock a RC (Resistenza-Condensatore) invece di circuiti controllati a cristallo che sono più stabili e precisi. Tuttavia, i cristalli possono qualche volta generare delle oscillazioni di terza armonica, causando in tal modo una frequenza di clock ben più alta di quella prevista. Inoltre, alcuni processori richiedono dei clock multifase e senza sovrapposizione, con temporizzazioni molto precise. In aggiunta, i livelli di tensione del clock non sono necessariamente compatibili TTL, ma possono avere una escursione di tensione maggiore. La specifiche del clock dei microprocessori si possono trovare nei data sheet (fogli di informazioni) relativi ai diversi dispositivi e possono essere controllate usando oscilloscopi e misuratori di frequenza di tipo convenzionale.

Reset all'accensione

Il circuito di azzeramento all'accensione (Power-Up Reset) del microprocessore può far sì che il sistema impazzisca, ma dopotutto «funzioni». Un impulso di reset non esistente, troppo breve, con dei disturbi o con fronti troppo lenti può far partire in modo sbagliato tutto il sistema, provocando degli azzeramenti parziali, disordinati o del tutto assenti. Si possono anche verificare dei problemi in quei circuiti di reset che sono sensibili agli impulsi spuri generati dall'alimentatore. Anche quando vengono usati dei circuiti ad ingresso di Schmitt, i fronti lenti possono causare, all'interno di alcuni sistemi, degli sfasamenti, da un dispositivo all'altro, nelle temporizzazioni di reset. Questo causerà l'accensione di alcuni dispositivi prima di altri, determinando un errato comportamento complessivo. Una sequenza troppo rapida di accensione-spegnimento-accensione (ON-OFF-ON) dell'alimentazione farà sì che molti sistemi non ripartano correttamente (è questo il caso del μLab). In questo caso potrebbe essere necessario aumentare il tempo di OFF in modo da permettere la scarica degli alimentatori e dei circuiti di riaccensione.

Nessuno di questi guasti al reset bloccherà necessariamente il funzionamento del sistema. Questo potrà funzionare per un breve intervallo di tempo e poi fermarsi, oppure bloccarsi all'interno di un loop di programma privo di significato, o anche effettuare la maggior parte delle sue operazioni. Il punto chiave da ricordare è che il sistema deve completare la sequenza di reset all'accensione per garantire che vengano eseguite tutte le operazioni di test, controllo e inizializzazione necessarie per far funzionare correttamente il sistema stesso.

I circuiti di reset all'accensione funzionano solo quando il sistema viene acceso. Questi circuiti possono essere analizzati, in quel preciso istante, mediante oscilloscopi a memoria, analizzatori logici e, in certi casi, analizzatori di firma. A scopo di prova, possono essere anche forzati manualmente e controllati dall'esterno.

Interruzioni

Delle linee di interruzione disturbate o bloccate possono essere causa di malfunzionamenti del sistema. Il sistema può funzionare con una linea bloccata ma in questo caso lavorerà molto lentamente (spendendo la maggior parte del suo tempo a servire l'interruzione «fantasma»). I disturbi sulle linee di interruzione possono generare alterazioni sporadiche del sistema o possono provocare, in istanti non richiesti, ingressi o uscite di dati da periferiche. Qualche volta il sistema non risponderà affatto ad alcuni dispositivi di I/O; questo può verificarsi per esempio quando una interruzione ad alta priorità viene a disabilitare quelle a più bassa priorità.

L'attività di una linea di interruzione può essere verificata con una sonda logica, con un analizzatore logico o con un oscilloscopio. Le interruzioni sono di natura asincrona e, a scopo di prova, possono essere controllate manualmente (abilitate o disabilitate).

Degradazione dei segnali

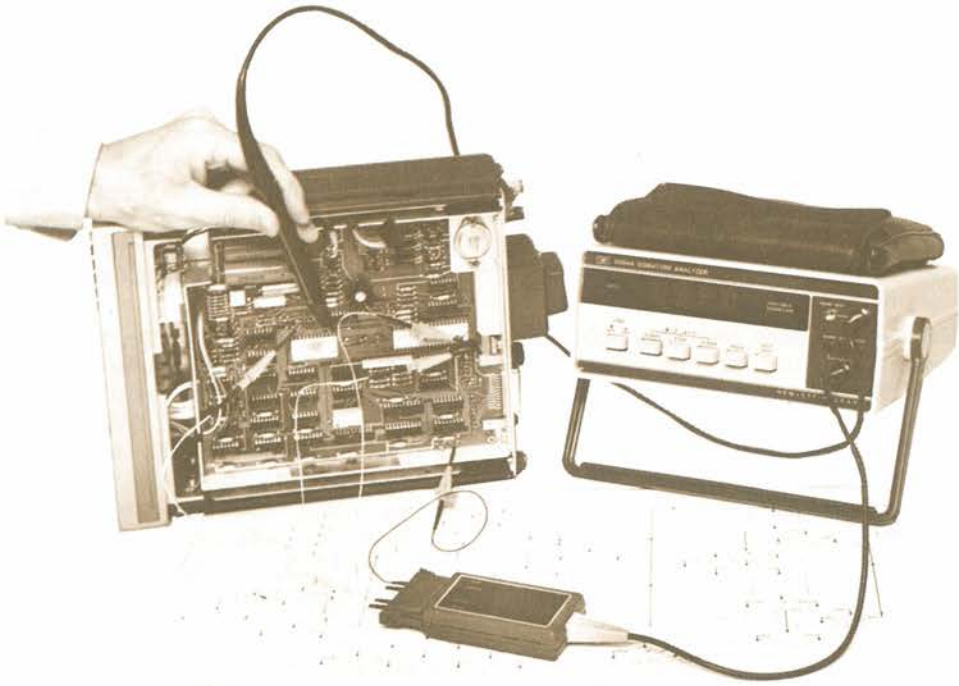
Le lunghe linee parallele dei bus e dei segnali di controllo, presenti nei sistemi a microprocessore medio grandi, sono qualche volta sensibili, su alcune linee critiche (come, per esempio, le linee di clock e di abilitazione) a problemi di interferenza (Crosstalk) e a quelli propri delle linee di trasmissione. Questi problemi possono manifestarsi come impulsi spuri su linee di segnale adiacenti o come oscillazioni su di una linea pilotata (determinando così transizioni multiple attraverso un livello di soglia logica). Entrambe queste situazioni possono portare a dati o a segnali di controllo sbagliati molto difficili da rivelare. Questo problema si verifica soprattutto quando le linee di segnale sono lunghe e introducono già dei ritardi apprezzabili nei segnali del sistema. Possono verificarsi dei malfunzionamenti quando vengono aggiunte, a questi sistemi, delle piastre di prolunga o quando esistono condizioni di elevata umidità. Gli accoppiamenti tra le linee poste sulle schede di prolunga possono originare un problema quando si hanno linee di segnale che corrono accanto ad altre linee in cui transitano segnali veloci (come per esempio nel caso di uscite di dispositivi Schottky), anche se si trovano su lati opposti di un circuito stampato.

Memorie

Memorie guaste, nei sistemi a microprocessore, possono provocare nel sistema parecchi tipi di malfunzioni. Si può verificare di tutto, dal guasto totale del sistema ad un semplice errore in un bit di un dato memorizzato. La maggior parte dei guasti alla memoria può essere trovata durante il programma autodiagnostico effettuato all'accensione, a meno che i guasti della memoria non impediscano anche l'esecuzione di questa prova. Se il sistema non prevede un test di verifica della RAM e se non vengono altrimenti fornite possibilità o procedure di test della RAM o ROM, è di fatto quasi impossibile controllare la RAM stessa. Quando una RAM diviene sospetta dovrete allora probabilmente ricorrere a tecniche di sostituzione.

I guasti alla RAM, che si verificano nell'area di memoria utilizzata dallo stack, causeranno generalmente un «crollo» generale del sistema, anche se l'errore si verifica su di un solo bit. Alternativamente i guasti alla RAM possono causare errori soft (morbidi, errori che non bloccano il sistema) con il risultato di un funzionamento non affidabile del sistema. Un altro fattore da considerare quando si devono diagnosticare delle malfunzioni relative alla RAM è costituito dai guasti localizzati nella circuiteria di rinfresco delle RAM dinamiche.

Anche le ROM possono essere guaste. Tali guasti sono più frequenti quando vengono usate le ROM non mascherate, ma programmabili. Un errore su di un singolo bit potrebbe bloccare il sistema, o, anche peggio, il 99 per cento del sistema potrebbe funzionare correttamente, mentre l'1 per cento potrebbe produrre risultati sbagliati. Le ROM possono essere efficacemente verificate durante l'autodiagnosi di accensione, posto che tali prove siano presenti nel sistema.



Utilizzo dell'analizzatore di firma per cercare un guasto in un prodotto basato su microprocessore

PROGRAMMI DI AUTODIAGNOSTICA

Diversamente dalle RAM però, le ROM possono essere provate anche con altre tecniche, nel caso il sistema non fosse provvisto di autodiagnostica. Una di tali tecniche richiede il funzionamento in free-run del sistema ed utilizza l'analizzatore di firma sia per verificare le firme riportate nella documentazione sia per confrontare le uscite di una ROM sospetta con quella di una ROM appartenente ad un analogo sistema funzionante (Esperimento 17.2).

La programmabilità dei sistemi basati sui microprocessori può essere usata in modo vantaggioso per contribuire al test dei sistemi. I programmi memorizzati nelle ROM possono provare e controllare le ROM, le RAM e lo stesso processore. Spesso è possibile provare in un qualche modo anche l'I/O. Il software può infine essere utilizzato per fornire degli stimoli ad uno strumento di test esterno come l'analizzatore di firma.

Test della ROM

La tecnica più comunemente usata per verificare le ROM fa uso di un *checksum* (somma di controllo). Quando viene programmata una ROM, si esegue la somma di tutte le parole in essa contenute ignorando i riporti risultanti. Questo numero viene complementato e memorizzato nell'ultima parola (o talvolta la prima) della ROM, così che quando tutte le parole vengono sommate insieme (incluso la parola di checksum memorizzata nell'ultima locazione) il risultato dovrà essere zero. Se il risultato non è zero alla fine di questa sequenza di test, vuol dire che all'interno della ROM esiste un dato sbagliato. (Nella realtà, il checksum viene generalmente calcolato in modo tale che il risultato totale sia un qualche numero diverso da zero).

Sfortunatamente, il checksum non è completamente affidabile. Esso rivela tutti gli errori dovuti ad un solo bit e la maggior parte di quelli causati da configurazioni multiple di bit; tuttavia, in molti casi, delle memorie contenenti due o più errori produrranno ancora un checksum corretto. Perciò, una ROM che passa il test di checksum è solo probabilmente buona. Se il test non funziona, sicuramente c'è qualche cosa che non va (sebbene potrebbe anche non essere la ROM a funzionare male).

Test della RAM

Le RAM vengono verificate scrivendo una configurazione di bit (Pattern) prestabilita nella memoria, rileggendola e verificando che questa non sia cambiata. Tra le molte configurazioni diverse che possono essere utilizzate, una molto diffusa è detta checkerboard. In questa configurazione, tutti i bit sono posti, in modo alternato, a uno e a zero. Una volta che sono state provate tutte le locazioni della memoria, il pattern viene ripetuto, con ciascun bit invertito rispetto a prima, verificando in tal modo che ciascun bit della RAM può memorizzare sia un uno che uno zero. Molte delle altre configurazioni, che vengono utilizzate per provare le RAM, sono particolarmente dedicate a rivelare i meccanismi di guasto all'interno delle RAM stesse.

Nessun test della memoria può dare una garanzia del 100 per cento, anche se fosse stato mostrato che ciascun bit può memorizzare un uno o uno zero. Le RAM possono essere sensibili alle configurazoni dei bit (Pattern Sensitive). Per esempio, una locazione potrebbe memorizzare correttamente 01010101 e 10101010 ma non funzionare quando viene memorizzato 01111000. Anche per una piccola RAM, occorrerebbe utilizzare un tempo estremamente lungo per controllare ogni possibile sequenza di uni e zeri. Per questa ragione, la credibilità del test RAM è generalmente molto più bassa di quella ottenibile nel caso delle ROM. Come per il test di checksum, una RAM che supera il programma di autodiagnostica del sistema è probabilmente buona. Se il test non viene superato, c'è invece sicuramente qualche cosa di non funzionante.

I/O IN MULTIPLEX

Le tastiere e i display, quando vengono utilizzati in multiplex, hanno spesso in comune alcuni dei circuiti di scansione (come avviene nel μLab). In queste situazioni, un tasto bloccato può sembrare un guasto del display. Analogamente, un ingresso guasto di un driver del display potrebbe causare un errore di tastiera. Nell'eseguire la diagnostica occorre perciò considerare l'interazione tra i circuiti comuni di scansione.

INTERFACCE

Molti sistemi a microprocessore si interfacciano con altri sistemi per mezzo di linee esterne di comunicazione (p. es., IEEE-488, RS-232C, modem telefonico). Queste linee sono generalmente lunghe e sono spesso esposte a sorgenti di interferenze elettriche: relè, trasformatori, motori, bobine e perfino impianti di illuminazione. L'interferenza elettromagnetica (EMI, Electromagnetic Interference) emessa da queste sorgenti può provocare una trasmissione sbagliata di dati, uno «sforzo eccessivo» (Overstressing) dei circuiti di interfaccia, e, specialmente nel caso dell'illuminazione, pesanti guasti ai componenti. Generalmente, i driver d'uscita delle linee tendono ad avere una frequenza di guasti più alta della media. Questo è dovuto sia alle interferenze EMI sia alle elevate correnti di commutazione che si generano nel pilotare i cavi capacitivi di interfaccia.

ALBERI DI RICERCA GUASTI

Un *albero di ricerca guasti* è un mezzo grafico utilizzato per illustrare la sequenza di prove che devono essere effettuate su un prodotto sotto esame. Questi alberi sono frequentemente disegnati come diagrammi di flusso, in cui il risultato di ciascuna prova determina quale sarà il prossimo passo da eseguire. L'uso degli alberi di ricerca guasti per riparare i prodotti basati sui microprocessori può far risparmiare una notevole quantità di tempo e fatica.

La Figura 18-1 mostra una parte dell'albero di ricerca guasti del Voltmetro Digitale 3455A. Teoricamente, per mezzo delle azioni effettuate e delle decisioni prese procedendo lungo l'albero, dovreste essere condotti a localizzare automaticamente il guasto nel prodotto. Sfortunatamente questo non si verifica sempre. Un albero di ricerca guasti perfetto dovrebbe considerare tutti i guasti possibili, un obiettivo questo difficile da raggiungere per la persona che deve definire l'albero di ricerca guasti. Anche gli alberi di ricerca guasti tendono ad essere largamente generalizzati, trascurando i dettagli utili per fare i test e prendere decisioni. Pochi alberi di ricerca guasti danno informazioni pratiche su come un test o una misura siano legati a ciò che il circuito fa o si suppone faccia. Se l'albero di ricerca guasti non riesce a condurvi direttamente al guasto effettivo, potreste trovarvi in un punto morto, senza alcuna idea di che fare come passo successivo. Tuttavia, l'albero di ricerca guasti sarà spesso la vostra migliore guida (almeno all'inizio).

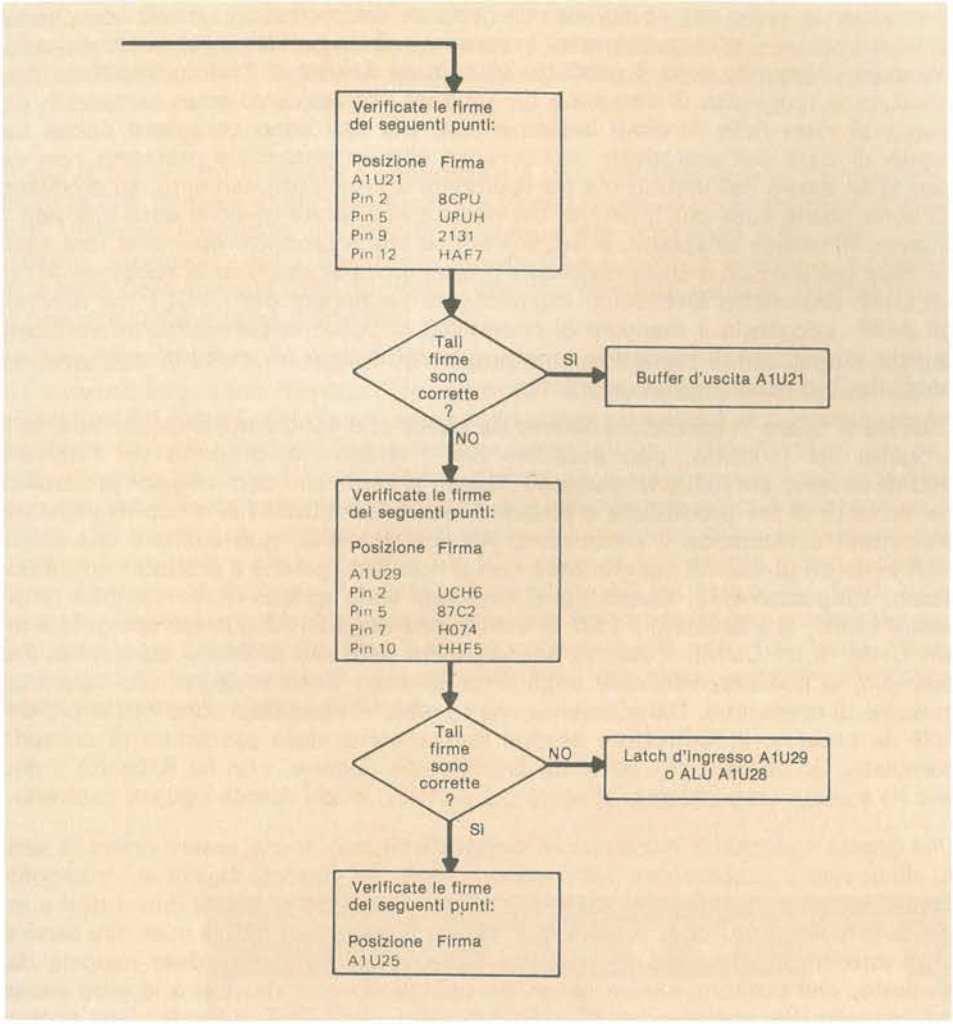


Figura 18-1. Albero di ricerca guasti tipico di un prodotto che preveda l'analisi di firma

Esistono alberi di ricerca guasti buoni e alberi di ricerca cattivi. Quelli buoni portano raramente ad un punto morto e forniscono una sequenza logica, e ben studiata, di test e misure, richiedendo un livello minimo di comprensione del circuito in prova. Spesso, per semplificare la procedura, implicano l'uso di tecniche avanzate, come l'analisi di firma. Nel ricercare i guasti di un prodotto, anche il più povero degli alberi di ricerca guasti può essere utile per localizzare nel sistema l'area del guasto e può permettere di risparmiare una considerevole quantità di tempo e di sforzi.

ALTRA DOCUMENTAZIONE

Per molti tecnici di assistenza e collaudo esperti, già i *diagrammi a blocchi* del prodotto possono fornire una buona quantità di informazioni utili a comprendere come le diverse parti del circuito interagiscano. La descrizione del funzionamento teorico del prodotto e l'albero di ricerca guasti corrispondente non sono in stretta relazione con l'hardware. Gli schemi invece forniscono spesso informazioni troppo dettagliate, rendendo difficile una visione «globale».

La parte rimanente di questa lezione evidenzia una serie, non vincolante, di passi generali che potete seguire per ricercare i guasti in un prodotto basato sui microprocessori. Vengono disseminate, all'interno delle descrizioni, numerose tecniche di manutenzione e «trucchi del mestiere».

ESISTE REALMENTE UN PROBLEMA?

È importante avere una comprensione generale del prodotto in modo che possiate essere sicuri di essere realmente in presenza di un problema. In qualche modo, dovreste conoscere cosa il prodotto fa e come funziona. I microprocessori permettono ai progettisti di disegnare dei prodotti che non solo sono complessi dal punto di vista delle funzioni implementate, ma che sono complessi anche dal punto di vista dell'operatività. Assicuratevi che un potenziale problema non sia perciò un errore dell'utente, ma sia realmente un malfunzionamento del prodotto. Ci sono poche cose più frustranti del cercare di riparare qualche cosa che non è guasto. In alcune situazioni, si verifica anche che il prodotto dovrebbe fare alcune cose per cui non è stato realmente progettato. Per esempio la selezione AC di un DVM (Voltmetro Elettronico Digitale) può funzionare per i VOLT ma non per gli AMP. Leggendo il manuale di operatività si possono generalmente verificare queste «limitazioni di progetto», che non costituiscono un malfunzionamento del prodotto, ma solamente un suo limite.

Quando si opera in condizioni diverse da quelle che sono state previste durante il progetto del prodotto, può accadere che i «bachi» di progetto del firmware (ROM) causino dei malfunzionamenti. Questi si verificano con maggior probabilità nei prodotti di pre-produzione e possono essere identificati nel modo migliore (se sospettati) contattando il costruttore. All'altro estremo, può capitare che esista realmente un problema ma che esso non si manifesti perché il prodotto non è utilizzato adeguatamente. Questi tipi di problemi sono spesso molto semplici da rivelare (p.es., si è bruciato il LED di indicazione degli OHM quando spingete il tasto OHM di un DVM). Possono anche essere però dei problemi complessi. Per esempio, si possono verificare degli errori quando viene eseguita una sequenza inusuale di operazioni. Dal momento che i problemi complessi sono molto più difficili da provare, il costruttore esegue normalmente delle procedure di collaudo complete. Un cliente, che porta un prodotto da riparare, non ha difficoltà a dire che ha trovato un problema. Diventa poi compito di chi ricerca i guasti risolverlo.

CHE COSA SI PUO' RICAVARE DAL PANNELLO FRONTALE?

Una grossa quantità di informazioni diagnostiche può anche essere ottenuta senza rimuovere il «coperchio» del prodotto. Molti dei prodotti basati sui microprocessori hanno un pannello frontale. Su questo ci potranno essere interruttori e indicatori, ingressi e uscite. *Spremere* (milking) il pannello frontale vuol dire servirsi degli interruttori, dei tasti e degli altri ingressi per sollecitare delle risposte dal prodotto, che possono essere osservate utilizzando gli indicatori e le altre uscite del pannello. Per esempio, se gli indicatori sono tutti spenti, quando viene acceso il sistema, potreste sospettare un guasto ad un interruttore, un fusibile, un cavetto di alimentazione, una connessione delle batterie o l'alimentatore. Se è rotto un segmento del display, il problema riguarda probabilmente il display stesso o il circuito che lo pilota. Se un DVM funziona male solo per la portata di 1-10 Volt, il problema può essere circoscritto ad una porzione relativamente piccola del circuito (l'attenuatore).

Cercate sempre di trarre il massimo beneficio da tutte quelle possibilità di verifica delle caratteristiche, di invio di messaggi di diagnostica e di test funzionanti all'accensione, che possono essere presenti nel prodotto. Tali possibilità sono indicate nel manuale del prodotto.

A questo punto potreste avere una certa idea di dove potrebbe essere localizzato il problema o potreste perfino averlo già individuato. Con buona probabilità, però, nulla di tutto questo si è verificato.

CHE COSA DICE IL MANUALE?

«Date un'occhiata al manuale solo se proprio non potete fare altrimenti». Questo atteggiamento piuttosto sbagliato (ma assai comune), nel caso di prodotti basati sui microprocessori, ha ancora meno significato di quanto non ne abbia per i prodotti convenzionali. Nel manuale si possono trovare molti aiuti e procedure di assistenza che vi aspettano per essere utilizzati. Il lavoro può essere reso molto più facile utilizzando speciali interruttori di collaudo, ponticelli, indicatori, possibilità e tecniche di test.

Cercate di capire i circuiti e di immaginare come stanno le cose. Controllate i capitoli del manuale relativi ai principi di funzionamento del sistema, i diagrammi a blocchi e gli schemi circuitali.

Non dovete fare tutto questo scendendo molto nei particolari ma quel tanto che basta per farvi un'idea di come vanno le cose. Identificate il microprocessore, la ROM, la RAM, gli I/O, il decoder degli indirizzi, il clock, i bus, le parti di controllo e di interruzione del sistema.

GUASTI IN PRODUZIONE E SUL CAMPO

La vita di un circuito integrato è generalmente una sequenza di eventi prevedibili. Nasce in una fabbrica di circuiti integrati e viene inviato ad un costruttore di prodotti. Viene poi inserito in una scheda, la quale a sua volta entra a far parte di un prodotto. Il prodotto entra finalmente in servizio e il circuito integrato ci rimane per il resto della sua vita utile. Superfluo dire che non tutti i circuiti integrati hanno una vita lunga e priva di malanni.

Gli utilizzatori di circuiti integrati stimano che circa il 2 per cento degli integrati che arrivano loro, sono difettosi. Un controllo, effettuato con un'apposita apparecchiatura (IC tester), dei circuiti integrati in arrivo permetterà di rivelare la maggior parte di quelli difettosi. Tale soluzione comporta un costo effettivo, per circuito integrato, di circa 10 centesimi di dollaro. Una volta che i circuiti integrati sono montati sulla scheda, il ritrovamento di quelli difettosi costerà invece circa 1 dollaro. Se questi non vengono rivelati prima che le piastre siano assemblate nel prodotto finito, sale a circa 10 dollari per IC il costo di una ricerca guasti e riparazione effettuata in fabbrica. La sostituzione di un IC sul campo (in field, cioè presso il cliente) è ancora più costosa: un costo tipico per trovare e sostituire un IC guasto in un prodotto che sia già presso un cliente è di circa 100 dollari. Chiaramente, diventa essenziale trovare ed eliminare il più presto possibile all'interno di tale sequenza gli integrati difettosi.

Il tester per circuiti integrati digitali HP 5045A viene utilizzato per effettuare la verifica dei componenti acquistati

Tipi di guasti

I tipi di guasti che si possono avere e le tecniche migliori per la ricerca degli stessi dipendono dalla storia del prodotto e dall'ambiente in cui viene provato. Quando in fabbrica viene collaudato per la prima volta un prodotto, quasi ogni cosa potrebbe essere causa di malfunzioni. I prodotti che si guastano quando sono installati hanno invece già lavorato tutti almeno una volta. Errori di assemblaggio (p. es., componenti inseriti male e collegamenti errati) non devono essere generalmente considerati tra i possibili guasti presenti sul campo. Anche la possibilità

di avere cortocircuiti dovuti a saldature o guasti multipli è molto più alta nel corso della produzione che a prodotto già installato. I guasti sul campo sono invece generalmente causati da componenti o connessioni che si sono guastati.

Tester automatici

A causa delle grosse quantità di prodotti identici che devono essere controllati all'interno della fabbrica, si possono giustificare tecniche e strumenti specializzati per la ricerca guasti e il collaudo. Tester (macchine che eseguono dei test) automatici di piastre e apparecchi di collaudo specializzati vengono spesso usati per ridurre il tempo impiegato per localizzare i guasti. In generale, è così possibile avere verifiche veloci, economiche ed accurate ed una diagnostica dei guasti. Tuttavia, non fatevi trarre in inganno da una eccessiva fiducia nelle prove automatiche delle schede, che vengono eseguite direttamente sotto il controllo di un calcolatore. Talvolta certe piastre, che sono state date come buone, sono in realtà difettose a causa dei limiti del tester (temporizzazioni, carichi o componenti) o del programma di prova che sta girando. Tuttavia, i più recenti tester per schede, che realizzano test dinamici, funzionali e parametrici più sofisticati, hanno aumentato notevolmente la credibilità di tali prove.

Il sistema di test schede HP 3060A permette di effettuare dei collaudi veloci, completi ed efficienti nell'ambiente di produzione

QUALI SONO LE COSE PIU SEMPLICI DA PROVARE?

Ha senso guardare per primo alle cose che possono essere verificate e riparate facilmente. Le cose semplici si possono guastare con la stessa probabilità di quelle complicate. Una cosa da guardare subito è l'alimentatore: all'interno di un prodotto, è generalmente una delle parti più portata a guastarsi. È anche una delle più semplici da controllare e generalmente facile da riparare. Tensioni fuori dai valori normali possono causare dei comportamenti imprevedibili del circuito. Si può impiegare molto tempo a cercare di trovare il problema se per prima cosa non viene controllata la tensione.

Può anche essere utile una ispezione meccanica. Circuiti stampati e connettori scadenti, fili rotti e parti mancanti possono essere generalmente trovati guardando o toccando i circuiti stessi.

TIPICI PROBLEMI DI RICERCA GUASTI IN PRODUZIONE

Nell'ambiente della fabbrica si possono trovare alcune cause comuni di guasti, semplicemente con una attenta ispezione visiva dei circuiti del prodotto assemblato. È facile così controllare che non ci siano degli interruttori e ponticelli messi male, dei componenti mancanti (sbagliati o invertiti) e dei punti di saldatura freddi. Un integrato di resistenze invertito può essere particolarmente difficile da diagnosticare elettricamente, dal momento che può causare interazioni tra dei nodi logici non in relazione, ma la sua mancanza può essere facilmente rilevata visivamente.

Due dei più comuni guasti di produzione sono i cortocircuiti dovuti ai processi di saldatura e doratura (ramatura) effettuati sulle piastre dei circuiti stampati. Questi possono essere di solito eliminati utilizzando un coltello tagliente. Quando non è conosciuta la posizione precisa del cortocircuito, esiste una tecnica per rimuover-

la piuttosto originale ma che spesso funziona. È anche utile in situazioni in cui la locazione del cortocircuito non è accessibile (come si verifica per i ircuiti tra gli strati interni in una scheda a più strati, Multilayer). Il procedimento richiede di caricare un condensatore da 100.000 μF (o più grosso) a cinque volt (una tensione sicura per i circuiti logici).

Poi, con dei cavi connessi ai due nodi in corto (attenzione alle giuste polarità), scaricate il condensatore e ascoltate il suono «secco» proveniente dalla piastra. Controllate la continuità per vedere se il corto si è aperto e, se questo non è avvenuto, riprovate ancora. Questa tecnica dovrebbe essere usata con cautela perché aprirà sempre il collegamento più debole posto lungo il percorso della corrente, che può non essere necessariamente la causa del guasto (potrebbe essere una pista sottile o un foro metallizzato). Il rivelatore di corrente fornisce un mezzo molto più sicuro per ritrovare i corti, come dimostrato nell'Esperimento 16-4.

Un problema relativamente nuovo che può verificarsi in produzione è quello che un piedino di un IC venga piegato male dalla macchina per l'inserzione automatica dei componenti. Queste situazioni possono far sì che non ci sia contatto elettrico tra il circuito integrato e il circuito stampato, o possono provocare dei cortocircuiti tra piste vicino o poste sotto il circuito integrato. Piedini piegati al sotto del componente sono spesso difficili da vedere perché potrebbero sembrare correttamente saldati nel giusto posto. Il modo migliore è verificare che sul lato inferiore della piastra spuntino tutti i piedini dei circuiti integrati o cercare di vedere, dal lato componenti della scheda, sotto i diversi circuiti integrati.

Sui circuiti stampati vengono spesso utilizzati connettori ricavati direttamente dallo stampato della piastra. Questi connettori possono causare dei problemi in produzione quando i loro bordi non sono perfettamente allineati o quando sono accidentalmente ricoperti con del solder resist o con dei collanti. Tali problemi possono essere rivelati da una semplice ispezione visiva.

Le schede a circuito stampato multistrato (multilayer) soffrono di tutti i problemi propri delle piastre solite, a cui si aggiungono alcuni problemi loro peculiari. Osservando la piastra controluce, spesso si può notare un errato allineamento o una contaminazione degli strati interni (che possono causare perdite o problemi alle alte frequenze). Dal momento che spesso è impossibile riparare gli strati interni, si può essere costretti a gettare via l'intero stampato.

Le piastre a «wire-wrap» possono avere dei piedini incurvati che causano dei corti. Altri problemi comuni in produzione possono essere dovuti all'aver montato, nell'estremità sbagliata di uno zoccolo a 16-piedini, un circuito integrato a 14 piedini, all'avere dei collegamenti errati, dei cortocircuiti tra i fili di piedini adiacenti e al sorgere di accoppiamenti (interferenze) tra segnali, causati dalla vicinanza di gruppi di fili.

GUASTI MECCANICI IN PRODOTTI GIÀ INSTALLATI

Il controllo visivo di un prodotto che si guasta sul campo può servire a rivelare diversi tipi di guasti: fili staccati, piste interrotte, IC ceramici o blocchi di resistenze incrinati, piedini wire-wrap piegati e connettori sporchi. Spesso, per individuare delle connessioni intermittenti o staccate o dei relay bloccati si può dare una «botta calibrata» a un lato del contenitore. Spesso può essere anche utile forzare meccanicamente le schede e i connettori (ruotandole o flettendole) per localizzare alcuni di questi problemi. Quando un prodotto è «D.O.A.» (Dead On Arrival, non funzionante all'arrivo) o si guasta in un ambiente fisico ostile potreste sospettare dei connettori posti sul circuito stampato. Potete anche cercare di risistemare tutti gli assemblaggi e togliere e reinserire tutte le schede, in modo da verificare che il problema non sia causato da un contatto scadente di un connettore. Si può utilizzare una gomma da matita per pulire i contatti sporchi dei connettori.

TECNICHE GENERALI DI RICERCA GUASTI

Sostituzione delle schede

Se una qualsiasi scheda di circuito stampato è facile da rimuovere e sostituire, e se si ha una scheda buona a portata di mano, potete provare a sostituirla (Swap). Quando in un prodotto ci sono due schede o due parti esattamente uguali, potete scambiarle l'una con l'altra. Il rischio che si corre effettuando tali

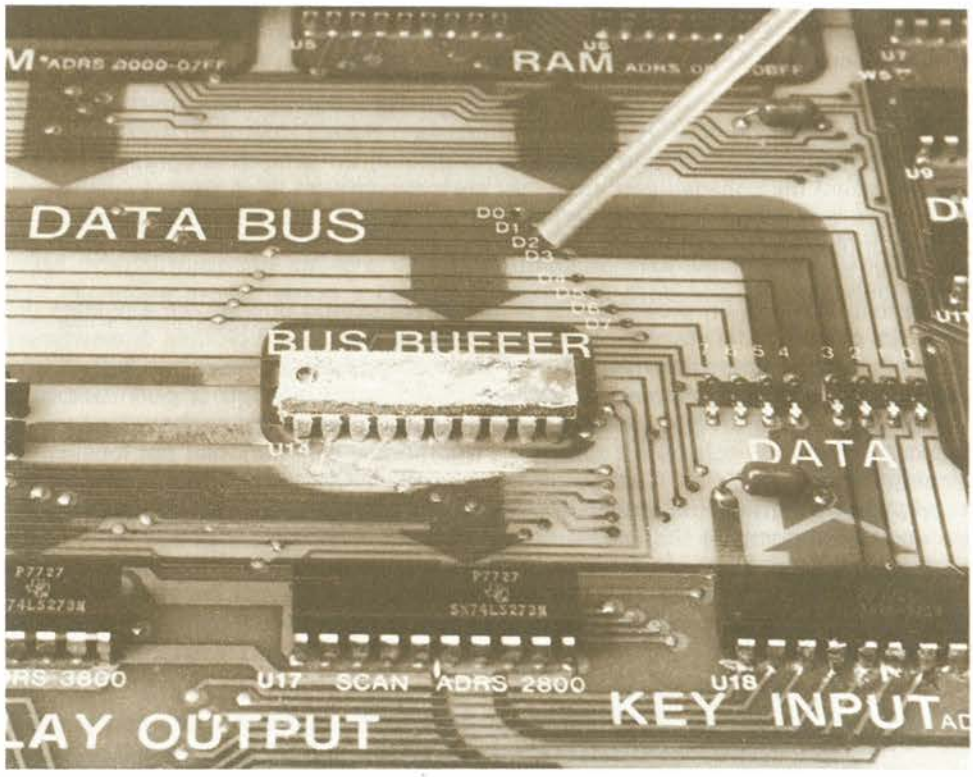
operazioni è quello di danneggiare una piastra buona essendo ancora presente lo stesso sovraccarico elettrico che ha danneggiato la precedente. In ogni caso, è necessario togliere l'alimentazione al sistema quando vengono rimosse o installate piastre o parti del sistema.

Se è disponibile un prodotto identico, può essere utile, qualche volta, un confronto funzionale. Questo confronto può essere utile specialmente in situazioni in cui non è chiaro se esiste effettivamente un problema hardware (potrebbe essere invece una caratteristica o una limitazione di progetto del prodotto).

Se il dispositivo sospetto è su zoccolo, per prima cosa provate a muoverlo per vedere se non si ha per caso una mancanza di contatto e cercate poi di sostituirlo con uno sicuramente buono. Notate, tuttavia, che uno degli ultimi dispositivi che dovreste sospettare, ma che in genere è il primo ad essere sostituito, è il microprocessore. La possibilità che un microprocessore sia guasto è in realtà molto bassa. Tuttavia, tali componenti sono spesso i primi ad essere sostituiti in una scheda perché sono complessi ed è difficile verificarne il funzionamento corretto. Lo stesso discorso vale per gli LSI che vengono usati con il microprocessore.

Test di stress

Per trattare guasti «marginali» o intermittenti può essere molto utile una tecnica chiamata *stress testing* (prova di sforzo). La prova di stress spesso può far sì che questi tipi di guasti siano temporaneamente evidenziati o attenuati: in entrambi i casi diventa più semplice identificare il guasto. Le schede vengono fisicamente sottoposte a sforzi di compressione o torsione, vengono sollecitate termicamente, scaldandole (getto d'aria o asciugacapelli) o raffreddandole (tramite uno spray freddo), ed elettricamente, modificando la tensione di alimentazione. Usando le prove termiche è possibile localizzare i dispositivi guasti all'interno di una piastra, con una precisione superiore a quella che si può avere con gli altri metodi, perché il calore o il freddo possono essere applicati direttamente ai singoli componenti. Fenomeni intermittenti possono essere dovuti a diverse cause: integrati critici, collegamenti interni agli integrati, giunture saldate, connessioni e circuiti di pilotaggio e temporizzazione.



Uno spray freddo vi può aiutare ad identificare dei componenti guasti o critici

Toccando i diversi dispositivi posti sulla scheda si può verificare se un componente si sta scaldando troppo (molto più caldo degli altri). Può darsi che siate in presenza di un problema, se trovate che un particolare dispositivo funziona ad una temperatura molto superiore a quella dei dispositivi dello stesso tipo.

Un dispositivo guasto può, qualche volta, essere talmente caldo da scottarvi il dito, per cui utilizzate con cautela questa tecnica. Fate attenzione al fatto che alcuni dispositivi buoni possono funzionare ad una temperatura maggiore di quella che vi aspettereste durante il normale funzionamento e che inoltre le temperature possono variare molto da un dispositivo ad un altro.

Cortocircuiti sull'alimentatore

Esistono alcuni modi efficaci per trattare i cortocircuiti dell'alimentatore. La prima cosa da farsi, in un sistema a più schede, è quella di cercare di localizzare su quale piastra sia il corto. Questo può essere fatto rimuovendo una piastra alla volta, fino a quando l'alimentatore non è più in corto. L'ultima piastra che è stata tolta è quella cortocircuitata.

Una tecnica per cercare poi il cortocircuito su una scheda guasta consiste nell'iniettare corrente nelle due linee cortocircuitate tramite un generatore di impulsi. Viene poi utilizzato il rivelatore di corrente per seguirla fino al corto. Ricordatevi che i condensatori (specialmente gli elettrolitici) assorbono un po' di corrente perché la corrente è di tipo impulsivo. Condensatori cortocircuitati possono essere rivelati utilizzando il rivelatore di corrente e confrontando i livelli di corrente che vanno in condensatori identici della stessa scheda. È molto probabile che sia cortocircuitato il condensatore che assorbe molta più corrente degli altri. Questa tecnica è particolarmente utile per trovare i cortocircuiti nei condensatori ceramici di bypass.

Un'altra tecnica utilizzata per localizzare i cortocircuiti del bus di alimentazione è quella di introdurre una corrente relativamente alta (circa 3 o 5 A) ai capi del cortocircuito. Assicuratevi di mantenere la polarità corretta della tensione e di non superare la tensione di alimentazione normalmente presente. Il percorso della corrente verso il cortocircuito può spesso essere determinato utilizzando un DVM con elevata risoluzione (0,01 mV), che misuri la caduta di tensione sulle piste del bus di alimentazione. Ai capi delle piste che si trovano lungo il percorso che porta al cortocircuito, e non altrove, si generano infatti delle cadute di tensione (Figura 18-2).

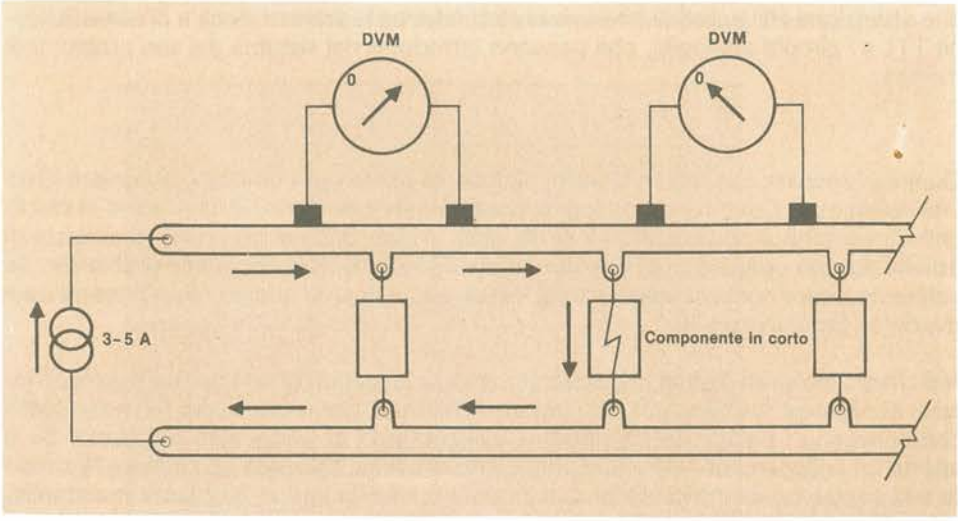


Figura 18-2. Uso di un voltmetro ad alta sensibilità per localizzare dei corti sulle linee di alimentazione

Una tecnica meno scientifica, ma molto più spettacolare, per localizzare i cortocircuiti sui bus di potenza consiste nel congelare l'intera piastra (a circa −10°C),

facendo sì che l'umidità si condensi su di essa. Alimentatela poi con circa 3-5 A. Mentre si riscalda e si scongela, il percorso della corrente diviene visibile e, in molti casi, viene evidenziato il cortocircuito.

COME PUO ESSERE ISOLATO IL GUASTO?

Una volta che sono provate, senza successo, le cose più facili, è giunto il tempo di affrontare i problemi «seri». A questo punto, l'abilità individuale di ritrovare i guasti, l'intuizione e la conoscenza del prodotto diventano effettivamente importanti.

Per prima cosa cercate di trarre il massimo vantaggio da tutte quelle caratteristiche che vi permettono di sezionare i circuiti, e che sono presenti nel prodotto e documentate: la possibilità di rimuovere una certa scheda, i ponticelli di servizio e ogni particolare modo o procedura di collaudo disponibile. Per permettervi di effettuare delle diagnosi indipendenti sulle diverse parti del circuito, può essere molto utile separare il sistema a microprocessore dai circuiti delle periferiche.

Un importante concetto della ricerca guasti è il cosiddetto *half-splitting* (taglia a metà). Sebbene il termine possa sembrarvi nuovo, probabilmente avete già usato questo metodo per anni senza neppure saperlo. L'half-splitting consiste nello scegliere un punto all'incirca nel mezzo del circuito. Rispetto a tale punto, il guasto si può trovare con uguale probabilità a monte o a valle. Se il sistema opera correttamente fino a tale punto, vuol dire che il guasto si troverà dopo di esso. Questo metodo funziona molto bene nel caso di circuiti che hanno dei percorsi di segnale «puliti» e unidirezionali, senza complicati anelli di reazione. Dal momento che i circuiti, che si trovano al di fuori della sezione con il microprocessore, sono spesso realizzati secondo tale struttura, anche i sistemi basati sui microprocessori possono utilizzare in modo positivo tale approccio.

In un tipico prodotto, il primo taglio è generalmente effettuato nel punto di interfaccia digitale-analogica. I circuiti analogici hanno la tendenza a rompersi più frequentemente di quelli digitali (poiché più «tirati» per quanto riguarda velocità, potenza, temperatura, sensibilità, regolazione, sovraccarichi esterni e ridotti margini di sicurezza dei componenti). Le caratteristiche di un prodotto sono frequentemente legate ai suoi circuiti analogici. Sono spesso questi circuiti che costituiscono la parte a «tecnologia avanzata» e che possono quindi trovarsi ad operare in condizioni prossime ai limiti. Inoltre, possono anche essere più numerosi dei circuiti digitali. Fate anche attenzione alle possibili interazioni elettriche tra le linee di clock e di alimentazione TTL e i circuiti analogici, che possono introdurre nel sistema dei seri problemi di rumore.

TIPI DI GUASTI DIGITALI

Quando i sospetti cadono sulla parte digitale, la prima cosa da fare è osservare l'attività dei segnali. Con una sonda logica potete esaminare l'attività dei segnali di clock, delle linee dei bus, delle abilitazioni dei chip, e delle linee di controllo. L'assenza di attività su uno qualunque di questi nodi sarà sintomo di un possibile problema. Se volete ripassare come si usa la sonda logica per la ricerca guasti, vi può essere utile rivedere l'Esperimento 16-1.

Nei circuiti integrati digitali, il guasto più tipico è la rottura di uno dei collegamenti interni al package (contenitore del circuito integrato). Sono questi dei fili molto sottili che collegano i piedini del contenitore alla piastrina di silicio vero e proprio. Se è aperto un collegamento interno a un piedino di uscita, tale piedino risulterà fluttuante e la sonda logica indicherà probabilmente un livello logico fluttuante ma stabile, perché a tale nodo sono collegati gli ingressi di altri dispositivi. Se è invece rotto il collegamento interno di un piedino di ingresso, si genereranno di solito dei malfunzionamenti su di una o più uscite di quel circuito integrato (che avrà qualche uscita bloccata a un livello logico alto o basso, o che eseguirà in modo non corretto le proprie funzioni logiche). Se una di queste uscite è collegata a un bus a tre stati, possono generarsi dei conflitti sul bus (più di una uscita presente sul bus allo stesso istan-

te) e, in questo caso, si può utilizzare il rivelatore di corrente per individuare il guasto. I conflitti sul bus possono anche essere osservati facendo uso di un oscilloscopio, a causa della presenza di livelli logici scorretti, ma ben definiti, sulle linee del bus; l'oscilloscopio, però, non fornisce alcuna informazione per quanto riguarda la causa del guasto (Figura 18-3). Anche linee di bus correttamente funzionanti possono del resto mostrare dei livelli cattivi, ma ben definiti: basta che siano spenti (cioè non attivi) tutti i dispositivi presenti sul bus.

Nei circuiti digitali, un altro tipo di guasto comune è il cortocircuito tra un piedino d'ingresso e una massa. Questo guasto è spesso causato dalla rottura del diodo di protezione dell'ingresso, posto sul chip. Si ha così un punto bloccato basso, che può essere osservato con una sonda logica. Un oscilloscopio, collegato ad un nodo che abbia questo tipo di problema, permette di osservare un livello di tensione, vicino a quello di massa, che cerca di salire (di pochi millivolt) tutte le volte che un'uscita collegata a tale nodo cerca di forzare un uno logico (Figura 18-4). Il rivelatore di corrente rappresenta un eccellente mezzo per individuare i piedini di ingresso cortocircuitati.

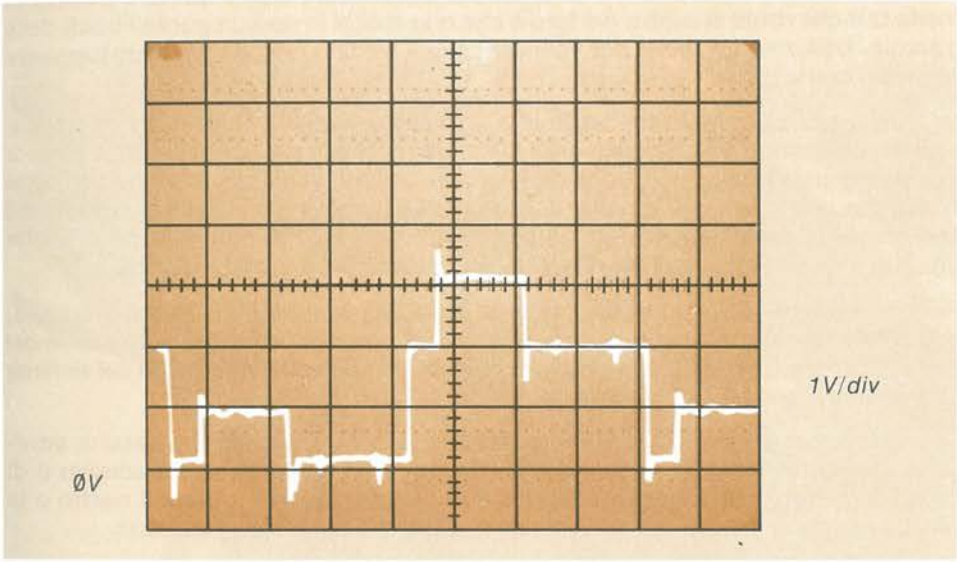


Figura 18-3. I conflitti sul bus determinano scorretti, ma ben definiti, livelli logici

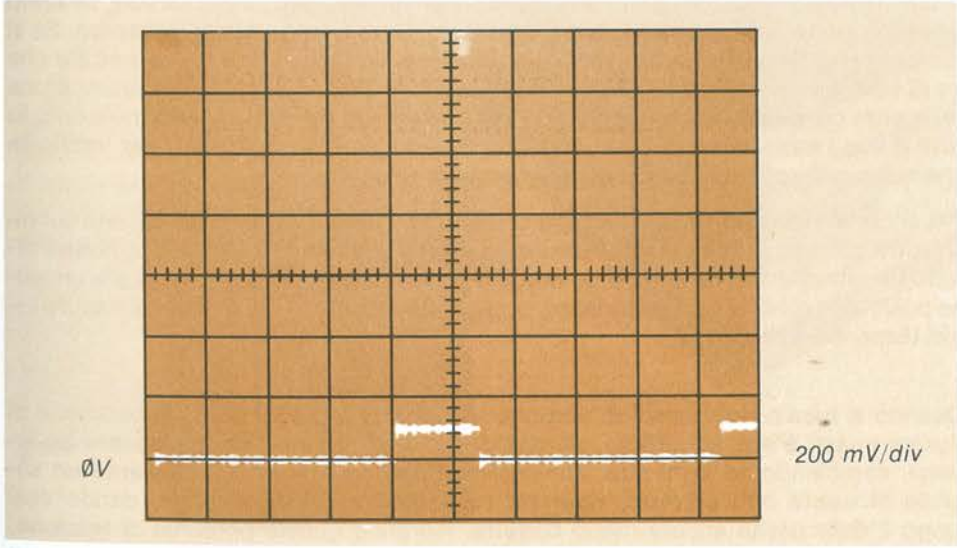


Figura 18-4. La rottura del diodo di protezione dell'ingresso, posto sul chip, determina un cortocircuito tra un piedino di ingresso e una massa

Se non è disponibile un rivelatore di corrente, un altro metodo che permette di rivelare delle uscite o degli ingressi bloccati è quello che prevede l'uso di un DVM di elevata risoluzione, e di un barattolo di spray freddo. Collegate il DVM al nodo bloccato e selezionate la portata di tensione DC più sensibile. Poi, mentre osservate l'indicazione della tensione, spruzzate, uno per volta, tutti i circuiti integrati che sono collegati al nodo bloccato, al fine di cambiarne la temperatura. Un qualsiasi cambiamento nella tensione del nodo (più di 10 mV) indica che il circuito integrato così raffreddato sta assorbendo corrente. Se non fosse disponibile uno spray freddo, potreste usare una sorgente di calore. Questa tecnica si basa sulla caratteristica, propria dei semiconduttori utilizzati negli IC, di avere tensione e temperatura in stretta relazione.

TECNICHE DI ISOLAMENTO

Una volta che si sia riusciti a concentrare i sospetti su di un particolare piedino di uscita o ingresso, è utile poterlo isolare dal resto del circuito. Una tecnica non distruttiva per effettuare questa operazione consiste nell'asportare lo stagno della saldatura dall'area posta tra il piedino e la scheda del circuito stampato, utilizzando un dissalatore ad aspirazione o una trecciola per dissaldare. Piegate quindi il piedino in modo tale che risulti al centro del foro e che non tocchi in nessun punto i bordi della piazzola. Utilizzate un tester per verificare che il piedino non sia più elettricamente collegato con la pista.

Le tecniche che potete usare per isolare i blocchi digitali di un sistema a microprocessore dipendono strettamente dalla struttura meccanica ed elettrica dello stesso. Può essere utile rimuovere alcune delle schede digitali, se tale operazione permette comunque, al nucleo del sistema, di operare. Se il nucleo è in grado di operare ad anello aperto, senza la reazione del bus dei dati, si può talvolta utilizzare il modo free-run, in modo da controllare l'attività del nucleo e del bus degli indirizzi.

Per interrompere certi segnali tra la scheda di circuito stampato e il resto del sistema, può essere usata una scheda di prolunga, che abbia degli interruttori sulle linee dei bus e dei segnali. In questo modo si possono aprire i percorsi di reazione del sistema e rimuovere le linee di bus bloccate.

Un modo ancor più semplice, per interrompere le linee di segnali che passino attraverso i connettori della scheda, consiste nel porre un pezzo di nastro adesivo o di carta sui contatti che si vogliono isolare. Per ricordarvi poi di togliere il nastro o la carta, è meglio che vi annotiate la scheda su cui avete fatto tale operazione.

Un metodo, ben poco convenzionale ma spesso efficace, per rivelare dei problemi sulle linee dei bus, è quello di misurare la resistenza verso massa (con alimentatore spento) di ogni linea di un certo bus (p. es., bus dei dati, bus degli indirizzi). Di solito le linee appartenenti allo stesso bus dovrebbero avere tutte la stessa resistenza. Se si rilevasse una differenza sostanziale per una certa linea, si potrebbe sospettare che su di essa è presente un problema. Se due linee invece mostrassero entrambe una resistenza più bassa di quella delle altre linee, potrebbe esserci un corto tra le due. In tutti e due i casi, prima di procedere, controllate gli schemi elettrici, per verificare che non sia il progetto stesso a giustificare tali differenze.

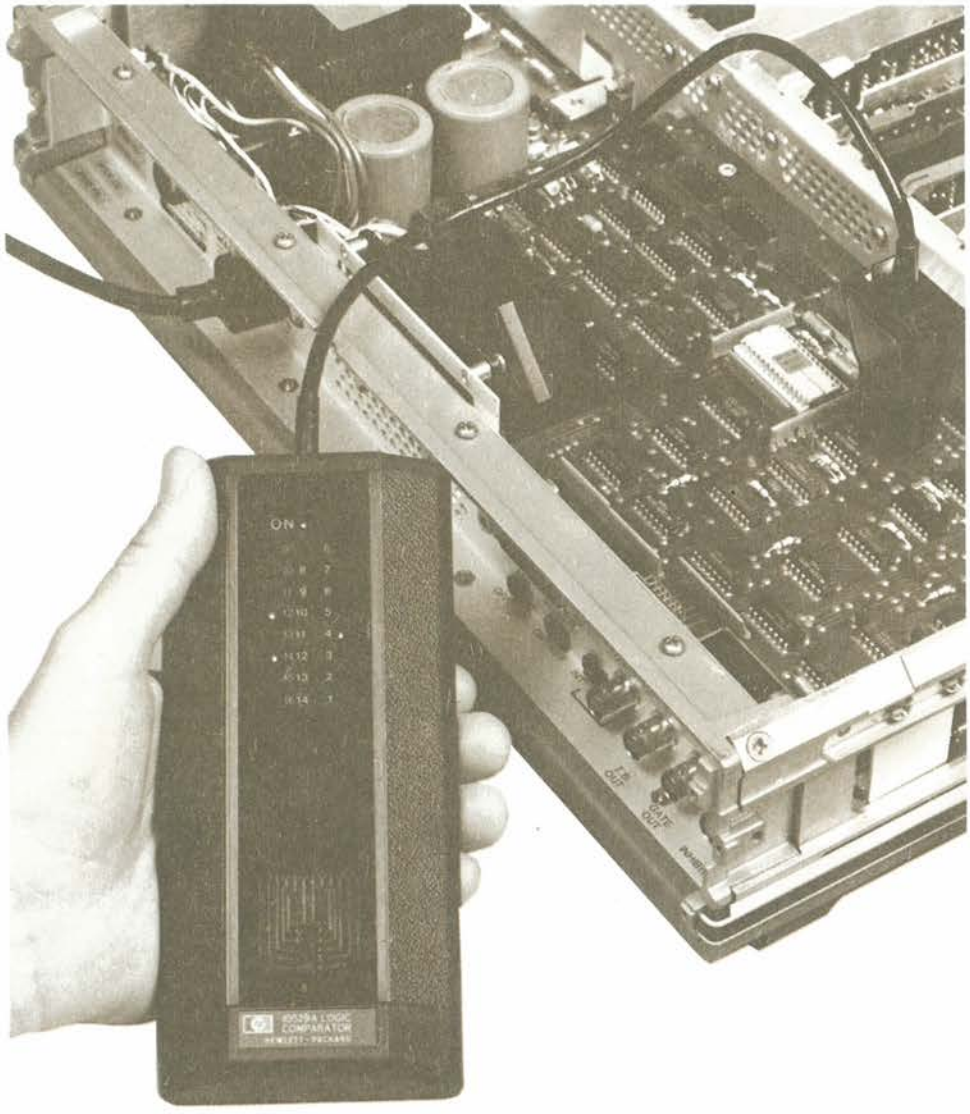
Per accertarvi che un circuito integrato stia operando correttamente, forzate sui dispositivi sospetti le linee di interruzione e i piedini di abilitazione del chip. Potete effettuare tale operazione cortocircuitando momentaneamente a massa, o alla tensione positiva, il piedino corrispondente, oppure utilizzando un generatore di impulsi logici (Esperimento 16-2).

ANELLI DI REAZIONE

Quando si hanno degli anelli di reazione (Feedback Loops) digitali, l'operazione di ricerca guasti è spesso difficile, perché gli errori si possono propagare e «rigenerare», cambiando ad ogni giro. Un anello di reazione, in cui sia presente un segnale di uscita non corretto, riporterà sull'ingresso questo segnale, dando così luogo a delle uscite ancora meno corrette. Aprendo questo percorso di reazione, si impedisce che dei segnali di uscita non buoni vengano riportati in ingresso. Se si possono introdurre poi nel loop dei segnali controllati, potrà essere osservato il flusso del segnale dall'ingresso all'uscita. Spesso, tuttavia, tale operazione non è

facile (potrebbe essere necessario controllare numerose linee). Del resto può anche essere difficile prevedere il comportamento corretto del circuito. Se è disponibile un altro prodotto funzionante (o una scheda con lo stesso circuito) è qualche volta utile fare sì che l'uscita del circuito buono controlli entrambi gli ingressi. In questo modo potete essere sicuri che il circuito sotto esame sta ricevendo in ingresso un segnale corretto. Si tratta poi di osservare i nodi dei due circuiti ed analizzare le differenze.

Un'altra tecnica, che può essere talvolta utilizzata per trovare degli IC guasti, è quella di montare un IC direttamente su di un altro (in inglese: piggy-backing, cioè lo stare a cavalcioni). Si tratta di osservare le uscite dell'IC sospetto con un oscilloscopio o con un analizzatore di firma; sopra questo IC viene poi posto un altro circuito integrato perfettamente uguale. I piedini, se necessario, dovranno essere leggermente piegati, in modo da essere tutti in contatto. Se i segnali cambiano, vuol dire che quel chip ha probabilmente dei problemi. Se invece non si osservano variazioni, e l'uscita non è bloccata, di solito si può allora supporre che il problema non stia in quell'IC. Fate attenzione ai circuiti sequenziali (contatori,



Comparatore logico di IC HP 10529A

registri a shift, ecc.) che possono dare delle uscite diverse a seconda delle diverse condizioni di partenza. Un modo migliore per realizzare questa prova è quello di utilizzare un comparatore di IC, come l'HP 10529A.

CONCLUSIONI

Nessuna conoscenza o esperienza può compensare totalmente una inadeguata documentazione di assistenza. In alcune situazioni il «tirare a caso» (sostituire componenti finché il problema non scompare) può essere la sola soluzione. Fortunatamente, la maggior parte dei prodotti basati sui microprocessori non dovrebbero spingervi in queste situazioni. I prodotti futuri comprenderanno probabilmente avanzate tecniche di assistenza, ad esempio l'analisi di firma, tanto più quanto più i progettisti si renderanno conto che i vecchi metodi e strumenti di ricerca guasti, utilizzati per la logica cablata, sono ben poco efficaci quando si ha a che fare con i microprocessori.

Talvolta la ricerca dei guasti produce risultati sconfortanti

I sistemi a microprocessore possono essere pensati come una estensione della logica digitale tradizionale. Sono invariati molti dei componenti, dei progetti circuitali e degli strumenti e tecniche di ricerca guasti. Tuttavia si impongono alcune significative differenze. I sistemi a microprocessore sono strutturati attorno ai bus, e molti dei circuiti collegati ai bus sono dei circuiti integrati LSI molto complicati. L'attività dei segnali tra i diversi dispositivi presenti sul bus è costante e complessa. È spesso utile aprire il bus dei dati, che costituisce il principale percorso di reazione del sistema, in modo che sia più facile isolare un guasto responsabile di un malfunzionamento dell'intero sistema.

Gli alberi di ricerca guasti, sebbene ci diano una metodologia ordinata per localizzare i guasti del sistema, non sono sempre sufficienti o adeguati a tale scopo. Esistono numerose tecniche, procedure e trucchi vari che possono rivelarsi efficaci al fine di diagnosticare e localizzare dei guasti nei prodotti basati sui microprocessori. Nel corso di questa lezione abbiamo visto numerose possibilità di questo tipo.

DOMANDE

Lezione 18

1. Nel caso dei bus del microprocessore, gli oscilloscopi sono meno efficaci, se si sta analizzando:
   a. un dato scorretto.
   b. un livello logico scorretto.
   c. dei problemi di temporizzazioni.
   d. dei conflitti sul bus.

2. Lo strumento più efficace per identificare il dispositivo difettoso su di un bus bloccato è:
   a. l'analizzatore di firma.
   b. l'analizzatore logico.
   c. l'oscilloscopio.
   d. il rivelatore di corrente.

3. Un potenziale problema degli alberi di ricerca guasti è che:
   a. sono difficili da seguire.
   b. hanno dei «bachi».
   c. richiedono una conoscenza approfondita del prodotto.
   d. possono condurvi ad un punto morto.

4. Il primo passo, di un'operazione di ricerca guasti, dovrebbe essere:
   a. leggere il manuale di assistenza del prodotto.
   b. controllare il fusibile e il cavetto di alimentazione.
   c. scuotere il prodotto ed ascoltare eventuali rumori.
   d. determinare la natura del problema.

5. Un requisito base, quando si vuole tagliare in due un circuito, è quello di:
   a. avere i percorsi dei dati unidirezionali.
   b. disporre dei modi di prova SA.
   c. effettuare la sostituzione della scheda.
   d. avere disponibile un prodotto campione per il confronto.

6. Nei circuiti integrati, il guasto più comune è dato dalla presenza di:
   a. un chip sbagliato all'interno del contenitore.
   b. un diodo d'ingresso cortocircuitato.
   c. un filo staccato all'interno dell'IC.
   d. un cattivo livello logico di uscita.

LEZIONE 19

Ricerca di guasti nel Microprocessor lab

INTRODUZIONE

Questa lezione vi presenta il diagramma di flusso per la ricerca guasti nel μLab e spiega come trovare un guasto utilizzando questo diagramma di flusso. Per vostro conto, potete inserire così dei guasti ed effettuare l'operazione di ricerca. Essendo un sistema a microprocessore completo, sul μLab si possono verificare la maggior parte dei guasti discussi nella precedente lezione. Perciò la strategia della ricerca guasti per il μLab è simile a quella utilizzabile in molti altri sistemi basati sul microprocessore.

IL DIAGRAMMA DI FLUSSO PER LA RICERCA GUASTI DEL MICROPROCESSOR LAB

La figura 19-1 presenta il diagramma di flusso per la ricerca dei guasti nel μLab. Vengono utilizzate le caratteristiche, contenute nello stesso sistema, di autodiagnostica e di analisi di firma. Questo diagramma di flusso fornisce un'utile guida per decidere le prove e le misure necessarie a localizzare un guasto circuitale.

C'è un problema?

La domanda non è così banale come potrebbe sembrare. Nel μLab si possono verificare molti guasti che non determinano dei malfunzionamenti evidenti. Alcuni esempi sono i guasti di accensione, l'impossibilità di memorizzare i dati in memoria, l'impossibilità di eseguire correttamente il programma dimostrativo ECHO memorizzato all'indirizzo 04D7, e guasti parziali alla tastiera.

Si accende?

Il μLab contiene un programma che verifica la condizione di accensione e che viene fatto partire ogniqualvolta viene acceso il sistema. Durante questa prova avviene l'accensione momentanea di tutti i LED di uscita e dei segmenti del display, l'altoparlante emette un «bip», e da ultimo, il display presenta il messaggio uLAbUP. Se ottenete tutti questi risultati, potete supporre (ragionevolmente) che vada tutto bene. Inoltre, da questo programma vengono eseguiti dei controlli sulla ROM e sui due chip di RAM. Se viene rilevato qualche problema, può essere visualizzato un messaggio, che fornisce il numero di identificazione del circuito integrato (IC4, 5 o 6) della ROM o della RAM eventualmente guasta. Il solo fatto che il μLab sia in grado di superare questa sequenza di inizializzazione e che controlli il display, vi dice che una buona parte del sistema a microprocessore funziona in modo sostanzialmente corretto (microprocessore, ROM, RAM, circuiti di decodifica e controllo, bus).

Si accendono le luci?

Se il μLab non si accende e nessuno degli indicatori luminosi a LED si illumina, si dovrebbero avere dei sospetti riguardo all'alimentatore o alla sorgente principale di alimentazione.

L'attività del bus?

Se l'alimentatore funziona, la prossima prova da fare è verificare la presenza di attività sulle linee del bus dei dati e degli indirizzi utilizzando una sonda logica o la sonda dell'analizzatore di firma. Una mancanza di attività dovrebbe spiegare la mancata accensione, che potrebbe essere causata da un guasto associato al clock. Potrebbe anche essere presente uno stato logico non corretto sugli ingressi di Reset, Hold o di Ready del microprocessore. Se queste linee sono a posto, o se una di esse è guasta, ma non si è in grado di dire perché lo sia, procedete con il prossimo punto. Anche una ROM guasta, che causasse l'esecuzione di una istruzione di HALT, potrebbe inibire l'attività del bus.

Funziona il loop di test SA?

Questo punto del procedimento è sostanzialmente un punto di «half-split» (taglio in due, tecnica descritta nella Lezione 18). Il loop di test per l'analisi di firma richiede che funzionino più circuiti di quanti sono necessari per il free-run. Cercate di porre il μLab nel loop di test SA premendo il tasto RESET e muovendo per una volta l'interruttore SA su e giù. Se il sistema sta funzionando, si accendono tutti i LED di uscita e tutti i segmenti del display e l'altoparlante genera un bip. Premendo il tasto INTRPT l'altoparlante emetterà ripetutamente dei bip. Se la maggior parte di queste azioni si verifica, probabilmente il loop SA funziona. Il modo migliore per esserne sicuri, è di collegare l'analizzatore di firma al μLab e controllare la firma letta per la Vcc con quella data nelle Tabelle C-3 o C-4 (il loop di test Sa di scrittura o di lettura).

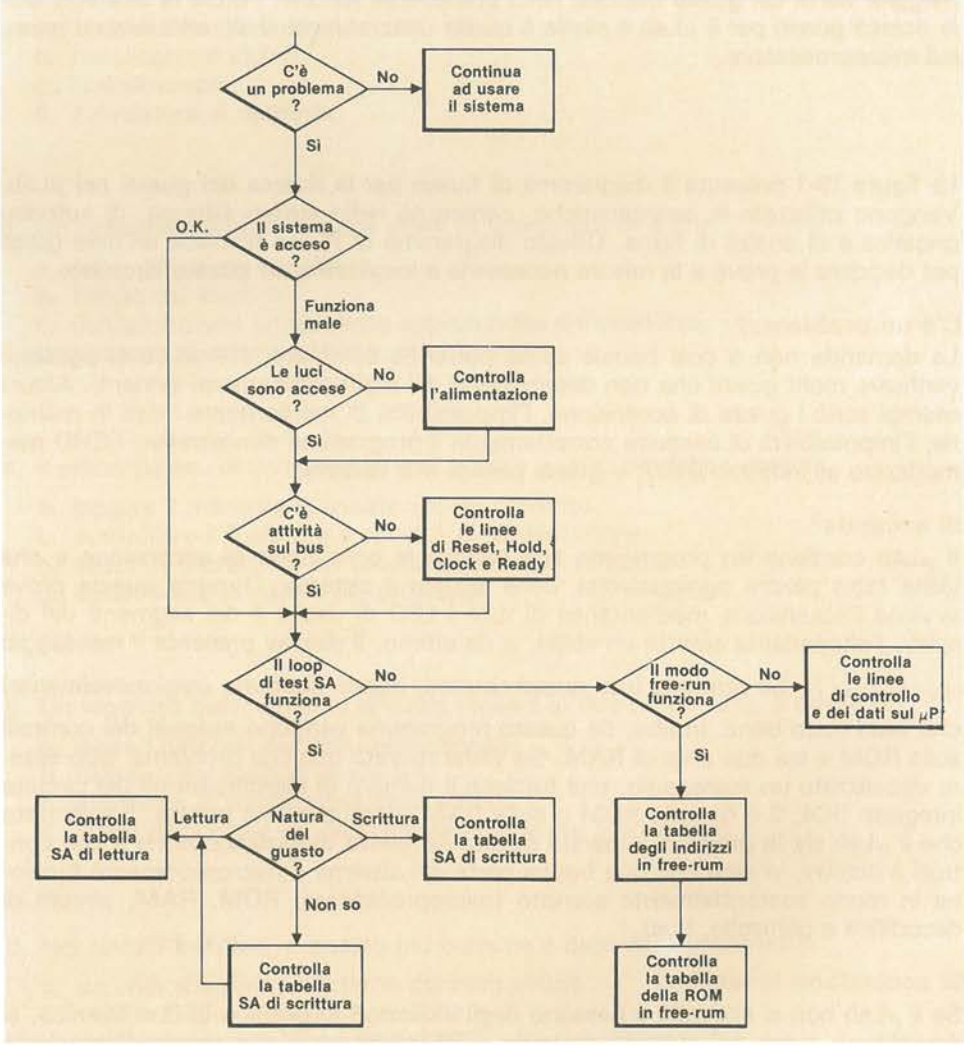


Figura 19-1. Diagramma per la ricerca dei guasti nel Microprocessor Lab

Se il loop di test SA gira, sapete che le parti essenziali del sistema stanno funzionando in modo soddisfacente. Il microprocessore indirizza la ROM, riceve le istruzioni della ROM per mezzo del bus dei dati, ed esegue queste istruzioni.

Nel loop di test SA ci sono due programmi di stimolazione. Uno di questi consiste nello scrivere, nel dispositivo con cui il microprocessore sta «parlando», delle configurazioni di stimolo; l'altro programma legge invece i dati dai dispositivi che il microprocessore sta «ascoltando». Nel loop di test SA entrambi questi programmi girano alternativamente. Si possono verificare sia i dispositivi che parlano sia quelli che ascoltano cambiando i collegamenti dell'analizzatore di firma.

I sintomi del guasto ci portano spesso ad una parte generale del circuito e sono classificati come problemi di scrittura o problemi di lettura. Per esempio, un display guasto è molto probabilmente un problema di scrittura. L'analizzatore di firma viene quindi connesso secondo la configurazione specificata nella tabella del loop di scrittura SA (Tabella C-3). Le firme rilevate sui nodi del circuito di display possono così essere confrontate con i valori riportati in Tabella C-3.

Funziona il free-run?

Quando non funziona il loop di test SA, dovete provare una parte di circuito più limitata. Aprendo le linee del bus dei dati dal lato del microprocessore (spostando verso l'alto gli otto interruttori del bus, BUS SWITCHES) e forzando un'istruzione di free-run (muovete verso l'alto l'interruttore FR), il microprocessore inizia, per proprio conto, a spazzolare tutto lo spazio degli indirizzi. Aprendo le linee del bus dei dati potete isolare il microprocessore dal bus dei dati e quindi dal resto del sistema. Nel modo free-run il microprocessore stimola le altre parti del circuito per mezzo del bus degli indirizzi. Questi stimoli portati dal bus degli indirizzi sono molto meno complicati di quanto non lo siano le configurazioni di dati, opportunamente controllati, che sono generate nel loop di test SA. Tuttavia il modo free-run è in grado di esercitare driver del bus degli indirizzi, i circuiti di controllo e di decodifica e le ROM. Il vantaggio di questo tipo di collaudo è che, per poterlo usare, è sufficiente che funzioni poco più del chip del microprocessore.

Il modo di test free-run può essere riconosciuto osservando il comportamento dei LED dei bus e di stato presenti sul μLab. I quattordici LED meno significativi del bus indirizzi (i più a destra) dovrebbero lampeggiare così rapidamente da apparire stabilmente accesi. I LED A14 e A15 dovrebbero invece lampeggiare rapidamente ma in modo visibile. I LED di stato dovrebbero comportarsi in questo modo: READ acceso, WRITE spento, ROM, RAM, INPUT, OUTPUT lampeggianti.

Se il microprocessore non riesce a funzionare neppure in free-run, verificate le linee di controllo ad esso collegate (Reset, Hold, Ready e Interrupt) ed il clock, in modo da vedere se la causa del guasto non risieda in una di queste. Potete controllare il circuito di inizializzazione all'accensione (Power-on Reset), forzando bassa la sua uscita. Potreste anche controllare i piedini del bus dei dati direttamente sul microprocessore in modo da vedere se sta ricevendo l'istruzione di free-run. Se tutti questi segnali sembrano corretti il microprocessore è probabilmente difettoso. Ci sono pochi altri fattori che potrebbero impedire ad un microprocessore, opportunamente alimentato e controllato, di funzionare in free-run.

Test degli indirizzi in free-run

Il test degli indirizzi in free-run vi permette di verificare la maggior parte dei circuiti di controllo e di indirizzamento del μLab. La Tabella C-1 mostra come deve essere disposto lo strumento e le firme di confronto. Una firma corretta di Vcc indica una corretta preparazione del test ed il corretto funzionamento in free-run del microprocessore.

Quando il μLab sta funzionando correttamente in free-run possono essere provati il bus dei dati e la maggior parte dei circuiti di controllo e decodifica. Le prime firme da guardare sono quelle delle linee del bus degli indirizzi. Esse dovrebbero essere tutte concordi con le firme della Tabella C-1. Possono poi essere rilevate le

firme sui piedini di selezione dispositivi, sul decoder degli indirizzi (IC7), sugli altri circuiti di controllo e sui piedini di controllo (IC3) del microprocessore. Se tutte queste firme sono corrette, potete supporre che le parti relative all'indirizzamento, alla decodifica e al controllo del sistema siano probabilmente a posto. Tuttavia, se continuate ad avere il sospetto che ci siano altri problemi, potete controllare le rimanenti firme della Tabella C-1. Se tutte queste firme sono corrette, dovreste sospettare che sia guasto un dispositivo presente sul bus dati e procedere con il test in free-run della ROM.

Test della ROM in free-run

Il test in free-run della ROM richiede una predisposizione dello strumento diversa da quella utilizzata nel test degli indirizzi, affinché i dati vengano campionati solamente quando viene indirizzata la ROM. In questo modo può essere verificato il contenuto della ROM e possono essere rilevate le linee guaste del bus dei dati. Entrambe queste condizioni, infatti, potrebbero impedire il funzionamento del programma di test SA del μLab (o di un qualsiasi altro programma).

La procedura, per provare la ROM, consiste nel collegare, per prima cosa, l'analizzatore di firma al μLab secondo le indicazioni della Tabella C-2 (mentre si è ancora nel modo free-run). Verificate tali collegamenti controllando la firma di Vcc. Controllate quindi ciascuna delle otto linee del bus dei dati. Una firma sbagliata su una di queste linee dovrebbe essere ricontrollata direttamente sul piedino di uscita corrispondente della ROM, in modo da capire se il problema è sull'uscita della ROM o su una pista della scheda. Se si trovassero delle firme sbagliate su tutte le linee del bus, dovrebbero essere controllati i piedini della ROM corrispondenti ai segnali di abilitazione ROM e agli indirizzi. Se queste firme sono valide, allora può verificarsi che sia erroneamente abilitato un altro dispositivo sul bus dei dati, e che si generi quindi un conflitto sul bus. Per questo problema, si possono controllare le firme sui piedini di abilitazione degli altri dispositivi posti sul bus. Se queste firme sono buone il sistema può essere fermato mentre la ROM è abilitata. A questo punto si possono utilizzare il generatore di impulsi ed il rivelatore di corrente per trovare il dispositivo che disturba il bus (fate riferimento all'Esperimento 16-4).

ESPERIMENTO 19-1

Familiarizzazione con la procedura di ricerca guasti

INTRODUZIONE

In questo esperimento, inserirete ad hoc un guasto nel μLab e seguirete l'albero di ricerca guasti, facendo osservazioni e misure e prendendo decisioni, fino a quando non troverete il guasto. Nel procedere al ritrovamento di questo guasto, acquisirete familiarità con gli strumenti e le tecniche usate per ricercare i guasti nel μLab.

PROCEDIMENTO

I. Simulazione del guasto

A) Prima di inserire il guasto verificate che il μLab sia privo di guasti. Per fare questo, spegnete semplicemente il sistema, aspettate un secondo e riaccendetelo ancora. Il display mostra uLAb UP e voi sapete quindi che non esistono grossi guasti.

B) Localizzate il ponticello W1 di guasto nella parte centrale in alto della scheda (riferitevi alla Figura 19-2). Osservate che è posto tra i due fori di sinistra di una serie di tre fori. Rimuovete il ponticello e reinseritelo poi nei fori di destra. Osservate che il display si spegne o visualizza sempre una sola cifra. Dimenticatevi per il resto di questo esperimento di avere mai toccato il ponticello W1.

C) Ora avete finito. Lo avete tolto e non potete ricordare che cosa avete fatto, giusto? Procedete nella lettura per vedere che cosa dovete fare a questo punto.

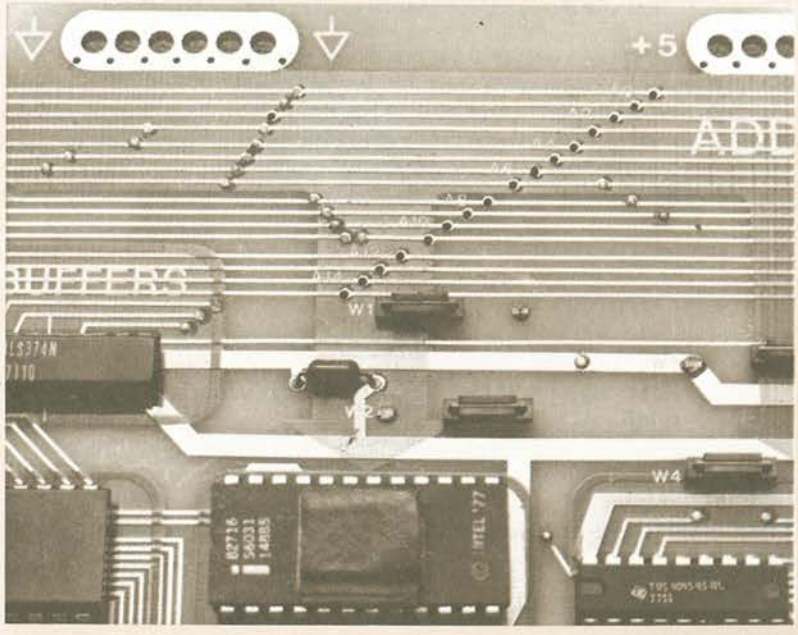


Figura 19-2. Posizione del ponticello W1 di guasto e fori di test per la sonda sul bus degli indirizzi

ESPERIMENTO 19-1

(continuazione)

II. Diagnosi del guasto

A) Il μLab possiede delle caratteristiche di inizializzazione, autodiagnostica, e diagnostica; provate perciò a spegnerlo e poi a riaccenderlo ancora. Forse, visualizzandovi un messaggio, vi dirà che cosa c'è che non funziona.

B) Non siete così fortunati. Premete il tasto [RESET]. Ancora niente. Premete qualche altro tasto. Ora iniziate a capire che si tratta di una cosa seria. Ci sono degli indicatori accesi? Sì. Questo significa che l'alimentatore è probabilmente funzionante. I bus dei dati e degli indirizzi stanno facendo qualche cosa? Sì. Non fate sempre affidamento sui LED, posti su questi bus, per accorgervi della loro attività. Non potete vederli lampeggiare se la frequenza è molto superiore a 30 Hz, e tale valore rappresenta del resto una frequenza decisamente bassa per un microprocessore.

C) Ora è giunto il momento di utilizzare la sonda logica. Collegate i cavetti di alimentazione di questa ai fori di alimentazione presenti sul μLab, posti sul lato alto della scheda (riferitevi all'Esperimento 16-1). Il puntale dovrebbe illuminarsi. Ponete l'interruttore a slitta della sonda sulla posizione TTL.

D) Portate il puntale sul foro A0 di prova per la sonda (Figura 19-2). Il puntale lampeggiante vi dice che esiste un'attività logica sulla linea A0. La sonda converte la veloce attività (il flusso di dati) rivelata dal puntale in una frequenza sufficientemente lenta perché la possiate osservare.

E) Per vedere se c'è l'attività sul bus, analizzate con la sonda logica le altre 15 linee di indirizzo e le linee D0-D7 dei dati.

F) Dal momento che esiste attività su tutte queste linee, voi sapete che nessuna di esse è bloccata alta o bassa. Questo vi dice che il microprocessore sta cercando di fare qualcosa.

G) È chiaro che il microprocessore non può far funzionare l'intero sistema, ma forse potrà essere eseguito il programma di loop di test per l'analisi di firma, inserito nel μLab. Cercate di far girare questo programma muovendo l'interruttore a slitta SA del μLab verso l'alto e poi verso il basso. Se si sono accesi tutti i segmenti del display e i LED di uscita e se l'altoparlante ha generato un bip, vuol dire che il loop SA è stato messo in funzione. Nel nostro caso però non è stato generato né un suono né un lampeggio. Proseguite la lettura, in modo da vedere che cosa significa questa situazione.

VERIFICA 1

Punti negativi:

- L'intero sistema non funziona.
- Il display non fa niente di utile, né sembra rispondere ai tasti.
- Il programma di test SA non gira, anche se esso richiede che solo pochi circuiti del sistema operino.

Punti positivi:

- Essendoci una certa attività del bus, voi sapete che il microprocessore sta funzionando (in una certa misura).

- L'alimentatore sembra essere a posto. Un rapido controllo sulle sue tensioni effettive dovrebbe confermarvelo.

Conclusione: Il problema sembra essere collocato in una zona tra il microprocessore e qualcosa connesso al bus. Sfortunatamente, questo coinvolge la maggior parte del circuito.

III. Suddivisione del sistema

A) Iniziate isolando il microprocessore. Aprite il bus dei dati. È il solo grosso percorso di reazione del sistema. Per aprire il bus, spostate verso l'alto (posizione FREE-RUN) gli otto interruttori del bus posti nel contenitore DIP, giusto sotto il microprocessore (indicati con BUS SWITCH). Con questo si scollega il microprocessore da tutti gli altri dispositivi posti sul bus dei dati e si permette allo stesso di funzionare ad «anello aperto».

B) Adesso che il bus dei dati è aperto, il microprocessore dovrebbe essere posto nel modo free-run, che gli permette di passare ciclicamente ed in continuazione attraverso tutto il campo indirizzi. Muovete verso l'alto l'interruttore indicato con FR (vicino all'interruttore SA) in modo da forzare nel microprocessore, da hardware, un'istruzione che lo obbliga ad operare in free-run.

C) Verificate con la sonda molte linee dei dati e osservate che i LED delle linee di indirizzo di ordine più alto lampeggino visibilmente, onde verificare il funzionamento del modo free-run. Questa attività del bus non dovrebbe sorprendervi, poiché viene fatta funzionare una parte del sistema minore di quella utilizzata prima.

VERIFICA 2

Che cosa avete fatto?

- Rinunciato a far girare un programma nel sistema.
- Isolato il nucleo del sistema a microprocessore dalla maggior parte del resto del sistema, permettendogli di funzionare in modo ad anello aperto free-run.

Che cosa potete fare ora?

- Potete esaminare il microprocessore da solo.
- Potete vederlo passare in modo ciclico attraverso il campo indirizzi.
- Potete valutare quale sia l'effetto del bus degli indirizzi sulla ROM, sul decoder degli indirizzi e su altri circuiti.
- Potete esaminare il bus dei dati trascurando il suo effetto sul microprocessore dal momento che non esiste più alcuna istruzione o percorso dati che ritorna al microprocessore (reazione).

Quali strumenti occorrono per fare quanto detto sopra? Ricordatevi che avete bisogno di analizzare le lunghe e complesse sequenze di dati che sono presenti nei modi logici di questi dispositivi. Il ciclo di free-run è lungo 2^{16}, cioè 65.536 cicli!

ESPERIMENTO 19-1

(continuazione)

IV. Esame dei nodi logici

A) Uno di tali strumenti è l'analizzatore di firma, discusso nella Lezione 17. L'analizzatore di firma rende facile dire se un nodo è buono o no, comprimendo una lunga sequenza di complessi dati seriali in una «firma» di quattro cifre.

B) Collegate l'analizzatore di firma come mostrato in Figura 19-3 e accendetelo. Predisponete, come mostrato, gli interruttori sul pannello frontale.

C) La sonda dell'analizzatore di firma può essere usata anche come sonda logica, se si osserva la lampada posta sul puntale. Toccate il punto di massa (↓) con la sonda. Osservate che sul display si legge 0000, come dovrebbe essere sempre quando la sonda è collegata a massa. Ora toccate Vcc con la sonda. Sul display dello strumento si dovrebbe leggere il valore 0001. Se non fosse vero, controllate i collegamenti e la posizione degli interruttori sul μLab (interruttori FR e DATA BUS verso l'alto) e sull'analizzatore di firma. Se vi sembra che non ci sia attività sul bus, premete il tasto [RESET].

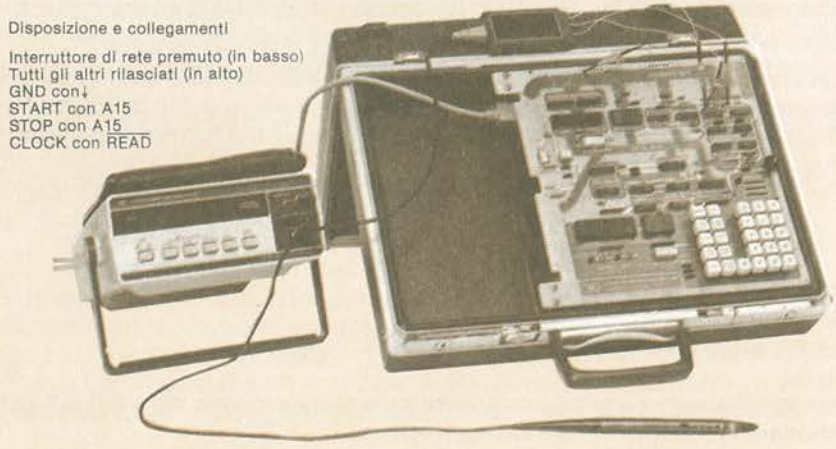


Figura 19-3. Come preparare l'analisi di firma per il test degli indirizzi in free-run

VERIFICA 3

Che cosa ci dice la firma osservata?

- Che l'analizzatore di firma è predisposto e collegato nel modo giusto.
- Che il microprocessore sta funzionando correttamente in free-run ad anello aperto, perché il bus dati non è più collegato ad esso. Il nocciolo del sistema sta funzionando.
- Ora possiamo usare l'analizzatore di firma per controllare le firme sui nodi logici stimolati dal microprocessore in free-run.

V. Lettura delle firme

A) Consultate la Tabella C-1 e rilevate le firme su tutte le linee degli indirizzi, utilizzando i fori previsti per la sonda. Tali valori dovrebbero coincidere con quelli riportati in tabella.

B) Risulta che la linea di indirizzo A11 è guasta perché ha la stessa firma di A10. Che cosa significa questo? Potrebbe essere che queste due linee siano cortocircuitate?

C) Se non avete ancora aperto lo schema elettrico del μLab, fatelo ora. Notate che lo schema è disegnato in modo da riprendere la disposizione dei circuiti sulla scheda del μLab. Adesso localizzate sullo schema i punti collegati ad A10 e A11.

D) Osservate che A10 e A11 non sono direttamente connessi al microprocessore ma che tra quest'ultimo e le due linee di indirizzo vi è il buffer U1. La firma errata potrebbe essere causata da un cortocircuito tra A10 e A11 presente o a monte o a valle di U1.

E) Controllate ora le firme sui piedini 4 e 2 di U1 (gli ingressi dei buffer di A10 e A11), confrontandole con i valori corretti riportati sulla Tabella C-1. Essi sono corretti.

F) Per vedere se U1 è guasto controllate le firme sui piedini 5 e 3 di U1 (le uscite dei buffer). Anche queste sono corrette. Il guasto deve trovarsi tra le uscite di U1 (piedini 5 e 3) e i punti di test A10 e A11 del bus degli indirizzi.

G) Seguite le piste con la sonda dell'analizzatore di firma, muovendovi da U1-3 verso il punto di prova A11. Questo dovrebbe portarvi al ponticello guasto W1. Per il momento lasciate il guasto come sta.

H) Il guasto è stato trovato e non è necessario rilevare delle firme addizionali. Tuttavia, per soddisfare la vostra personale curiosità, guardate le firme sugli altri nodi indicati nella Tabella C-1 e vedete quali sono influenzati dal guasto di A11.

Prima di rimettere a posto il ponticello guasto W1, potete utilizzare questa situazione, di cortocircuito tra le linee del bus A10 e A11, per verificare l'utilizzo del rivelatore di corrente in tandem con il generatore di impulsi.

VI. Seguire la corrente

A) Collegate i fili di alimentazione del rivelatore di corrente e del generatore di impulsi ai fori di alimentazione posti sulla scheda del μLab.

B) Premete il pulsante del generatore di impulsi e spostatelo in avanti. Il puntale dovrebbe lampeggiare rapidamente, indicando che si trova nel modo 100 Hz. Se non fosse vero, spostate indietro l'interruttore, rilasciatelo e ripetete l'operazione. Il generatore è ora predisposto ad iniettare impulsi di corrente nei nodi.

(continuazione)

C) Toccate con il puntale del generatore di impulsi la linea A11, come mostrato in Figura 19-4. Il generatore, ora, sta iniettando impulsi di corrente in questa linea di indirizzo.

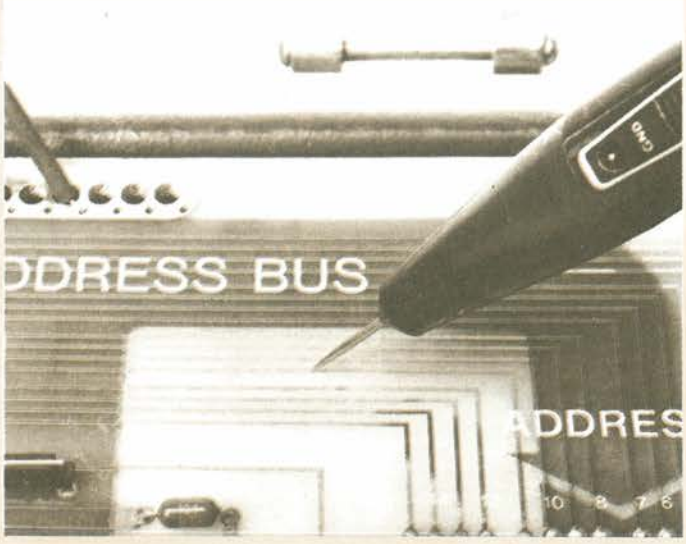


Figura 19-4. Impulsi di corrente sulla linea A11 degli indirizzi

D) Mantenendo il generatore d'impulsi sul nodo A11, ponete il puntale del rivelatore di corrente ad angolo retto, rispetto a quello del generatore, e toccatene così il puntale (osservate la Figura 19-

Figura 19-5. Regolazione della sensibilità del rivelatore di corrente

5). Assicuratevi che la manopola di regolazione posta sul rivelatore di corrente sia allineata (parallela) al puntale del generatore. Ora, regolate la manopola posta sul rivelatore di corrente, in modo che la lampadina si accenda solo debolmente.

E) Il rivelatore di corrente è ora regolato per rivelare la corrente iniettata nel nodo dal generatore d'impulsi. Muovendo il rivelatore di corrente, seguite la corrente dal puntale del generatore fino alla linea A11. Una volta che il rivelatore è in contatto con il circuito stampato della scheda, determinate in quale direzione va la corrente (a sinistra o a destra del puntale del generatore di impulsi). Ricordatevi che più la lampadina è luminosa, più è intensa la corrente. Assicuratevi di tenere i due fori, posti sui lati della punta, allineati con la pista A11 del circuito ed il puntale stesso disposto ad angolo retto rispetto alla scheda.

F) Forse, speranzosi, avete scelto di muovervi verso sinistra e avete trovato che, una volta che siete arrivati al punto mostrato in Figura 19-6, la lampada si offusca misteriosamente. Come sapete, la corrente non sparisce nell'aria né cade attraverso un foro della piastra. Perciò deve essere passata, attraverso i fori metallizzati, dall'altra parte della scheda.

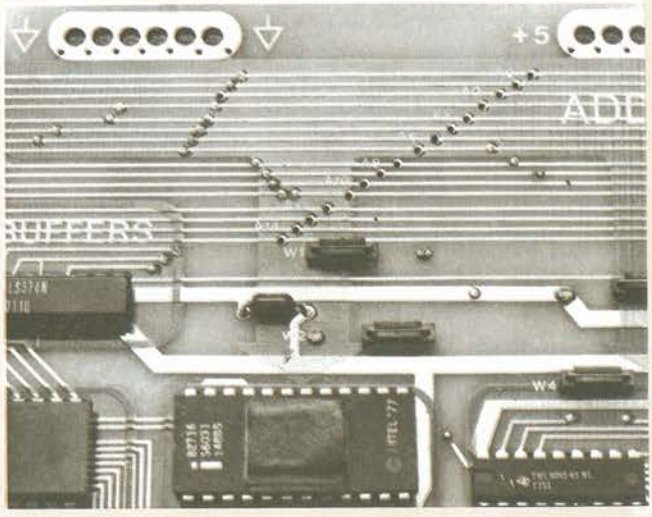


Figura 19-6. Punto dove cambia la corrente

G) Controllate quanto detto, alzando la scheda e continuando a seguire la corrente sul lato posteriore. Tirando, in modo deciso, la manopola nera posta vicino al centro del bordo destro della scheda del μLab, la piastra verrà sbloccata e potrete così ruotarla. Se farete in modo che la scheda ruoti fino alla posizione verticale, rimarrà poi ferma da sola.

H) Il rivelatore di corrente dovrebbe portarvi al ritrovamento del ponticello guasto W1, posto sulla linea A10. Ora avete risolto il problema.

I) Mettete il ponticello guasto 1 nella sua posizione originale (i due fori di sinistra). Forse desiderate verificare le firme sulle linee A10 e A11.

ESPERIMENTO 19-1

(continuazione)

J) Spostate verso il basso l'interruttore di free-run (FR), per riportarlo nel modo normale e fate la stessa cosa per gli interruttori del bus dei dati.

K) Premete una o due volte il tasto RESET ed osservate il messaggio visualizzato uLAb UP.

L) Spegnete e riaccendete per verificare che l'autodiagnostica di inizializzazione funzioni correttamente e che fornisca lo stesso messaggio.

RIASSUNTO

In questo esperimento è stato introdotto, mediante un ponticello, un guasto. Per eliminare alcune delle possibili cause di guasto sono stati direttamente osservati gli indicatori luminosi del μLab, ci si è avvalsi delle caratteristiche di autodiagnostica interna e si è fatto uso della sonda logica. Dal momento che nessun programma poteva girare, è stato aperto il percorso di reazione più importante (bus dei dati). Isolando il microprocessore dal resto del sistema ed evitando perciò l'interazione tra le due parti di sistema così create, si è predisposto poi il microprocessore nel modo free-run. Nel loop di test free-run è stato utilizzato l'analizzatore di firma al fine di identificare i nodi logici funzionanti e non funzionanti. Una volta trovate le linee guaste e determinata la natura del guasto (linee A10 e A11 cortocircuitate insieme), l'analizzatore di firma vi ha portato al guasto vero e proprio. Anche se il guasto era già stato trovato, sono stati utilizzati il rivelatore di corrente ed il generatore di impulsi per mostrare come anche questi strumenti avrebbero potuto portarvi all'identificazione del guasto. Il generatore ha iniettato impulsi di corrente in uno dei nodi, mentre il rivelatore di corrente è stato utilizzato per seguire la corrente verso l'altro nodo. Il guasto si trovava nel percorso che congiungeva i due nodi. Su un tipico prodotto, questo tipo di guasto è spesso causato da un «baffo» di stagno o da ponticelli di doratura e qualche volta può essere molto difficile identificarlo con una osservazione ad occhio nudo o per mezzo di tecniche che si limitino ad analizzare delle tensioni.

RICERCA DEI GUASTI CON UN OSCILLOSCOPIO

Anche un comune oscilloscopio può essere utilizzato per localizzare questi tipi di guasti. Il procedimento è simile a quello utilizzato con l'analizzatore di firma. La differenza fondamentale sta nell'interpretazione delle misure.

Nell'operazione di identificazione del guasto descritta in questo esperimento, l'oscilloscopio può essere utilizzato per verificare la presenza generale di attività sui bus degli indirizzi, dei dati e di controllo e sui piedini di abilitazione dei dispositivi. Facendo questo, dovreste anche rilevare, sulle linee A10 e A11 del bus, delle forme d'onda identiche. Tuttavia, a meno che queste linee non fossero già sospette, sarebbe poco probabile notare questa coincidenza.

Se il sistema viene fatto funzionare in free-run, l'oscilloscopio può visualizzare, una dopo l'altra, le sedici linee del bus degli indirizzi. Dal momento che la frequenza del segnale su ciascuna delle linee di indirizzo dovrebbe essere doppia o metà di quella delle due linee adiacenti, risulterà evidente sullo schermo il cortocircuito tra A10 e A11 (Figura 19-7).

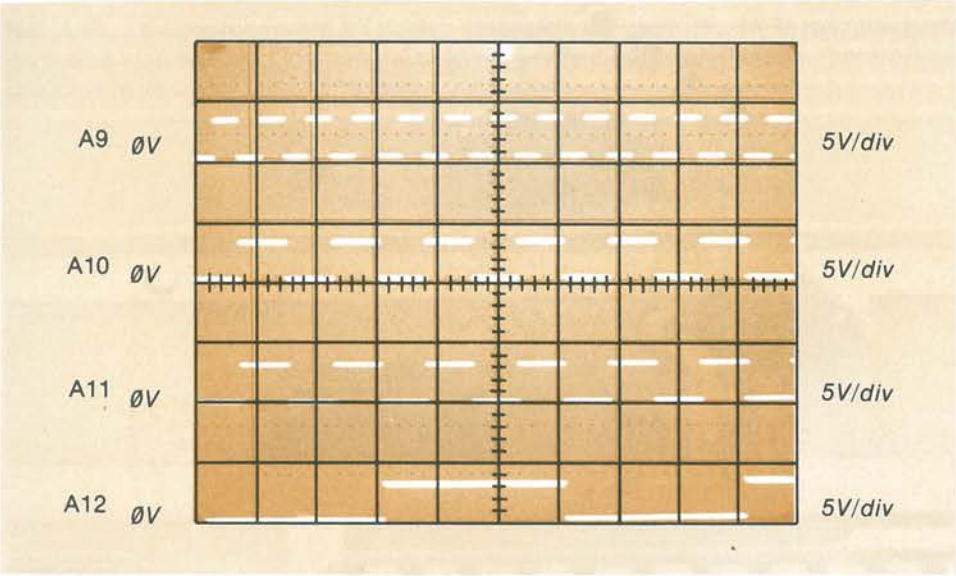


Figura 19-7. Schermo dell'oscilloscopio con A10 e A11 in corto

Mentre l'analizzatore di firma dà informazioni del tipo passa/non passa, le forme d'onda dell'oscilloscopio devono essere analizzate in funzione dei risultati attesi. È molto difficile, con un oscilloscopio, valutare le lunghe e complesse sequenze di bit che sono presenti nei sistemi basati sui microprocessori. Tuttavia, possono essere subito identificate delle malfunzioni nei driver, nelle temporizzazioni e dei guasti funzionali grossolani (p. es., nodi bloccati).

ALLA RICERCA DEI GUASTI

Il prossimo passo è quello in cui voi stessi dovete trovare i guasti. Tutte e dodici le locazioni dei guasti sono poste su degli zoccoli a tre piedini con un ponticello che collega il piedino centrale con uno di quelli esterni (Figura 19-8). Una posizione del ponticello è corretta, mentre l'altra introduce un guasto nel circuito. I ponticelli non sono riportati sullo schema. I ponticelli devono sempre essere installati in una delle due coppie di fori. Sulla facciata inferiore della scheda, è presente un punto accanto alla posizione normale (non di guasto) del ponticello. Utilizzate questi punti come «ancora di salvezza» per riportare il μLab al suo modo normale di funzionamento.

Questi ponticelli per guasti simulano dei malfunzionamenti circuitali che si possono trovare nella realtà: piste cortocircuitate o aperte, uscite bloccate o aperte e guasti funzionali all'interno dei circuiti integrati. In alcuni casi, uno solo di questi ponticelli

fa sì che si verifichi in realtà più di un cambiamento. Per esempio alcuni guasti aprono una traccia e la cortocircuitano con un'altra per impedire che si verifichino dei sovraccarichi distruttivi sui componenti.

Se da un lato questi ultimi tipi di guasti, a differenza degli altri tipi, non si verificano spesso nella realtà, sono tuttavia estremamente istruttivi (allo scopo di imparare ad individuarli).
Potete introdurre da voi un ponticello di simulazione di guasto o avere qualcuno che lo fa per voi (in modo che voi non sappiate quale sia). Per quelli che amano la difficoltà, ricordiamo che è anche possibile introdurre dei guasti multipli.

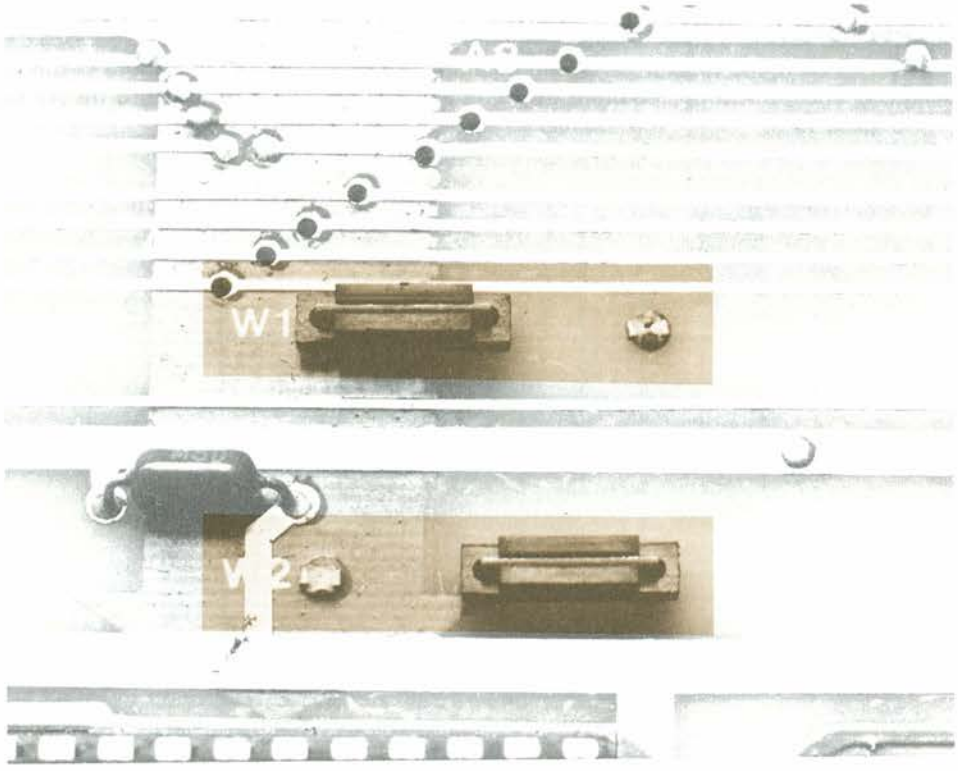


Figura 19-8. Tutti e dodici i ponticelli di guasto hanno due possibili posizioni: normale e guasto. Sul retro della scheda, un puntino indica la posizione normale (non guasto)

Come accade per i guasti presenti nei circuiti reali, alcuni guasti sono più facili da ricercare di altri. Se vi trovate in difficoltà con un particolare guasto, potete pensare di lasciarlo da parte e di ritornarvi dopo che ne avete trovati altri, quando avete accumulato una maggiore confidenza in questi problemi. Non abbiate paura di andare a vedere le soluzioni se siete realmente in difficoltà. Potete imparare molto di più dalle soluzioni che arrancando in modo inutile. Le soluzioni vi possono mostrare il punto in cui avete preso una decisione sbagliata, fatto un'osservazione sbagliata, o dimenticato di effettuare una misura. Dopo che siete sicuri di avere trovato il guasto potreste voler confrontare le vostre soluzioni con quelle documentate nell'Appendice A.

Le descrizioni delle soluzioni al problema dei guasti sono date in due modi:
- Il percorso semplificato, indicato sul Diagramma di Ricerca Guasti del Microprocessor Lab
- Una descrizione più dettagliata del procedimento di ricerca guasti, che utilizza anch'essa il Diagramma di Ricerca Guasti del Microprocessor Lab.

Entrambe le soluzioni fanno uso dell'analisi di firma.

CARATTERISTICHE ATIPICHE DEL μLAB

Ci sono alcune differenze rilevanti tra i tipi dei guasti simulati nel μLab e quelli che potrebbero verificarsi su altri prodotti basati sui microprocessori. Nessuno dei guasti del μLab è marginale, causato da componenti guasti, connettori, alimentatore o difficile da individuare visivamente. Tutti i guasti sono localizzati su una sola scheda di circuito stampato ed inoltre il μLab contiene un numero relativamente piccolo di dispositivi e nessun circuito analogico.

Ricordatevi che il μLab è uno strumento didattico. Un certo numero di caratteristiche, presenti in esso a tal fine, non sono presenti invece nella maggior parte dei prodotti basati sui microprocessori. Non si può pensare che su altri prodotti ci sia un libero uso di LED, che indicano le funzioni del bus e del controllo, come si verifica nel caso del μLab. Inoltre, la possibilità di introdurre manualmente dei programmi e di operare passo passo è una caratteristica assai poco comune quando si hanno applicazioni dedicate. Ancora, la facilità con cui è possibile individuare la disposizione dei dispositivi e delle linee del bus ed il libero uso di segni grafici posti sul circuito stampato del μLab non sono compatibili con le limitazioni economiche e di dimensioni proprie di strumenti più complessi ad indirizzo non educativo.

Nel μLab, il microprocessore è il nucleo essenziale del prodotto. Nella maggior parte delle altre applicazioni, l'identità dei prodotti è definita dalle periferiche, mentre il microprocessore opera puramente come controllore.

RIPASSO

Lezione 19

Questa lezione è stata completamente dedicata al problema della ricerca dei guasti sul μLab. È stato discusso l'albero di ricerca guasti del μLab, siete stati guidati a trovare una soluzione al problema della presenza di un guasto reale. Sono stati descritti i ponticelli per simulare i guasti e vi è stata data la possibilità di individuare voi stessi dei guasti.

Lezione 19

1. Se non funziona il loop del test free-run:
   a. anche il loop SA non funziona.
   b. può essere bloccata una linea di controllo del microprocessore.
   c. il microprocessore può essere guasto.
   d. potrebbe essere vero tutto quanto detto ai punti a, b, c.

2. Il vantaggio principale del loop di test SA rispetto al modo di test free-run è che:
   a. è più facile che funzioni.
   b. richiede meno circuiti per funzionare.
   c. rileva più velocemente le firme.
   d. opera su una parte maggiore di circuiti.

3. Se il μLab supera positivamente l'autodiagnostica di inizializzazione, si può dedurre che:
   a. l'intera piastra è priva di guasti.
   b. il bus degli indirizzi e dei dati sono privi di guasti.
   c. il nocciolo del sistema e la tastiera sono privi di guasti.
   d. la tastiera e il display sono privi di guasti.

4. Lo scopo principale di aprire le linee del bus dei dati, tra il microprocessore ed il resto del sistema, è quello di:
   a. ridurre il carico del bus sulle uscite del microprocessore.
   b. impedire che il microprocessore riceva istruzioni inviate dalla parte rimanente del sistema.
   c. impedire che il microprocessore invii dei dati alla parte rimanente del sistema.
   d. realizzare tutto quanto detto ai punti a, b, c.

5. Due linee, che danno una firma non corretta ma identica, possono essere:
   a. ingressi e uscite di un IC buffer.
   b. cortocircuitate verso massa.
   c. cortocircuitate insieme.
   d. tutto quanto detto ai punti a, b, c.

VI
UNA PANORAMICA DEGLI ALTRI MICROPROCESSORI

Il μLab utilizza il microprocessore 8085 dell'Intel. Questa lezione descrive alcuni altri microprocessori largamente utilizzati. Pur essendo i concetti di base sempre gli stessi, i dettagli possono variare notevolmente da un microprocessore ad un altro. Ora che vi sono chiari i concetti base, sarà relativamente facile imparare a conoscere gli altri microprocessori.

LEZIONE 20

Panoramica dei microprocessori

L'8085 E GLI ALTRI MICROPROCESSORI

Il primo microprocessore a otto bit commercializzato nel mondo è stato l'Intel 8008, introdotto nel 1971. Rispetto alle caratteristiche attuali appare molto lento e difficile da utilizzare, ma al momento della sua introduzione sul mercato si presentò come un grosso salto qualitativo. Due anni dopo fu presentato l'8080 che rappresentava un miglioramento significativo per quanto concerneva sia la velocità che la facilità di realizzare attorno ad esso dei progetti. L'8080 divenne rapidamente uno standard industriale. L'8085 costituisce la terza generazione di questa serie. È più veloce dell'8080 e richiede un minor numero di componenti per realizzare un sistema completo.

L'Intel non rimase a lungo sola nel campo dei microprocessori. Il primo e principale concorrente dell'8080 fu il Motorola 6800. Il 6800 ha un certo numero di caratteristiche non presenti nell'8080, ma non è diventato altrettanto popolare.

Lo Z80 della Zilog è un altro discendente dell'8080. Costituisce un miglioramento sostanziale dell'8080, ma è abbastanza diverso dall'8085. Più oltre, in questa lezione, verranno descritti l'8080, lo Z80, l'6800 e altri importanti microprocessori.

I MICROPROCESSORI A QUATTRO BIT

I microprocessori a otto bit sono più potenti di quanto è richiesto in molte semplici applicazioni di controllo. I microprocessori a quattro bit, essendo più semplici e meno costosi, sono spesso più adatti a tali situazioni. I primi microprocessori a quattro bit richiedevano ROM, RAM e I/O esterni, esattamente come i processori a otto bit. I dispositivi più recenti comprendono, in un singolo circuito integrato, RAM, ROM e I/O.

Per quanto riguarda l'architettura, la differenza più significativa tra i microprocessori a otto bit e quelli a quattro bit sta nel fatto che i processori a quattro bit usano due memorie separate, una per i programmi e una per i dati. La memoria di programma (generalmente ROM) è di solito organizzata per parole di otto bit, mentre la memoria dati è su parole di quattro bit. Le parole ad otto bit sono usate per le istruzioni, dal momento che una parola a quattro bit permette solamente sedici diverse istruzioni.

I MICROPROCESSORI A SEDICI BIT

All'altro estremo dello spettro delle applicazioni, troviamo alcune situazioni in cui i processori a otto bit si mostrano insufficienti. I microprocessori a sedici bit, che sono simili come architettura interna ai minicalcolatori, sono in grado di dare una potenza di calcolo maggiore in queste applicazioni. Essi dispongono di capacità aritmetiche notevolmente migliorate e sono ben adatti ad applicazioni orientate ad elaborazioni numeriche. I microprocessori a sedici bit più recenti hanno delle caratteristiche estremamente sofisticate al fine di facilitare l'implementazione di sistemi operativi e di linguaggi ad alto livello.

Dal momento che i microprocessori a sedici bit possono trattare sedici bit per volta essi possono eseguire calcoli con numeri grandi più velocemente di quanto non possano fare i processori ad otto bit. Anche i processori ad otto bit possono trattare dei grandi numeri, ma sono costretti a suddividerli in parti di otto bit, rallentando così significativamente il calcolo.

MICROCALCOLATORI SU UN SOLO CHIP

Il passo successivo all'integrazione di un'intera CPU su un solo chip è stato quello di includere un po' di RAM, ROM e I/O in un unico circuito integrato. Questo è stato fatto per microprocessori a quattro, otto e sedici bit. Questi processori hanno una memoria di programma ROM compresa tra 1K e 4K o più byte, ed una memoria RAM per la memorizzazione temporanea dei dati da 64 a 256 o più byte. Le linee di I/O disponibili vanno da 16 a 32. Alcuni dei dispositivi più avanzati includono sul chip un UART (che controlla i trasferimenti seriali di I/O). Questi circuiti realizzano un microcalcolatore completo su un solo circuito integrato e costituiscono una soluzione semplice ed economica nel caso di molte applicazioni di controllo. Spesso con uno di questi IC si può sostituire un'intera piastra costituita da dozzine di SSI e MSI.

Alcuni microcalcolatori su un singolo chip (Single Chip Microcomputer) sono disponibili con UV EPROM al posto della ROM. Questo facilita la realizzazione di sistemi prototipo. Quando poi il progetto è completato, puè essere utilizzata una versione ROM meno costosa.

I PROCESSORI BIT-SLICE

I processori bit-slice (lett.: a fette di bit) differiscono in modo sostanziale dagli altri tipi di microprocessore. Ogni chip di un processore bit-slice ha una dimensione di parola relativamente piccola (generalmente quattro bit), ma è possibile unire in modo parallelo parecchi di questi elementi in modo da costruire una parola lunga quanto si vuole. Per esempio, usando quattro chip bit-slice da quattro bit si può formare un processore a sedici bit.

Un'altra caratteristica che distingue i processori bit-slice è data dal fatto che non hanno un insieme di istruzioni prefissate. L'utilizzatore deve scriversi il *microprogramma* che definisce l'insieme di istruzioni. Questo permette di adattare l'insieme di istruzioni alle diverse applicazioni, creando in tal modo dei programmi veloci ed efficienti. Tuttavia, questo aumenta notevolmente gli sforzi di progetto richiesti per costruire un sistema.

I processori bit-slice sono dispositivi bipolari, mentre gli altri processori di utilizzo generale (general-purpose) sono dispositivi MOS. Perciò i processori bit-slice sono molto più veloci e consumano più potenza. Essi richiedono anche l'uso di memorie veloci per evitare una degradazione delle prestazioni del sistema.

Come suggerisce questa breve discussione, i sistemi basati su bit-slice sono molto potenti ma anche molto complicati. Contengono generalmente dozzine di IC e la programmazione è piuttosto complessa. Vengono utilizzati in applicazioni che richiedono particolari caratteristiche di velocità e potenza, come quando si abbiano ad elaborare dei segnali in modo digitale o si vogliano dei processori aritmetici ad elevata velocità o dei sistemi di controllo molto potenti o quando si vogliano realizzare dei minicalcolatori.

Attualmente sono disponibili più di trenta tipi di microprocessori general-purpose e circa uno stesso numero di microcalcolatori a singolo chip. (Riferitevi alla bibliografia per maggiori informazioni sui dispositivi disponibili). In questo capitolo vengono descritti alcuni dei tipi più utilizzati: 8080A, Z80, 6800, la serie 6500, F8 e TMS 1000.

L'Intel 8080

La Figura 20-1 mostra la CPU base realizzata con l'8080, il predecessore dell'8085. Per realizzare le funzioni del solo 8085 l'8080 richiede tre chip: un generatore di clock (8224), una CPU (8080) ed un dispositivo per il controllo di sistema (8228). L'8080 non condivide (multiplex) le linee degli indirizzi e dei dati, ma presenta invece sulle stesse linee, in tempi diversi, i segnali di controllo ed i dati. Il circuito di controllo 8228 separa le informazioni di controllo. La massima frequenza di clock dell'8080 è 2 MHz, mentre quella dell'8085 è 3 MHz. (Come per la maggior parte dei microprocessori, sono disponibili per l'8080 e l'8085 versioni selezionate che possono operare a velocità più elevate). L'8080 richiede tre alimentazioni (+5, −5, +12V) mentre l'8085 richiede solo +5V.

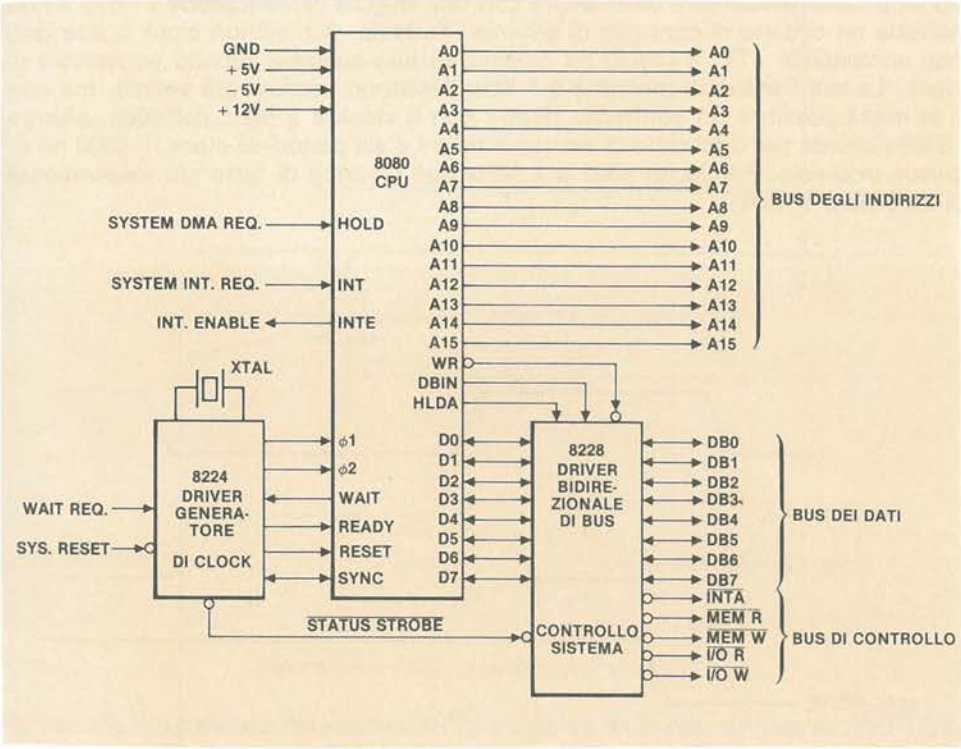


Figura 20-1. CPU in tre chip basata sull'8080

All'8080 mancano gli ingressi di interruzione RST 5.5, 6.5, 7.5 e l'ingresso TRAP. Inoltre non ha nessun piedino di I/O seriale. L'insieme di istruzioni è identico a quello dell'8085, che dispone però di due istruzioni addizionali per controllare le interruzioni e l'I/O seriale. Tutti i programmi dell'8080 possono girare anche sull'8085, ma le temporizzazioni saranno diverse. Quasi tutti i progetti nuovi, invece che utilizzare l'8080, utilizzano l'8085.

Lo Zilog Z80

Lo Zilog Z80, come l'8085, è un discendente dell'8080. Tuttavia ha molte caratteristiche non presenti né sull'8080 né sull'8085. Come l'8085, non richiede nessun circuito generatore di clock né un controllore di sistema e funziona con una sola alimentazione (+5V). Diversamente dall'8085, il suo insieme di istruzioni è stato notevolmente ampliato. Esso comprende tutte le istruzioni dell'8080 (in modo tale

da poter operare con i programmi dell'8080), ma comprende anche un gruppo sostanzioso di nuove e potenti istruzioni. Per esempio, si può muovere, con una sola istruzione, un blocco di dati lungo fino a 64K byte da una parte ad un'altra della memoria. Con una sola istruzione si può controllare, azzerare o porre ad uno un qualsiasi bit di un qualsiasi registro. Questi sono solo due dei molti nuovi tipi di istruzioni che possono considerevolmente facilitare il lavoro del programmatore. Sono stati aggiunti anche i modi di indirizzamento relativo ed indicizzato (descritti nel seguito a proposito del microprocessore 6800). Lo Z80 contiene tutti i registri dell'8080 (B, C, D, E, ecc.), ma tali registri sono tutti duplicati.

Questa descrizione, pur essendo lontana dall'illustrare completamente le prestazioni dello Z80, serve a mostrare un concetto fondamentale: lo Z80 è un processore del tipo 8080, con un certo numero di semplificazioni hardware ed una capacità software notevolmente migliorata.

Il Motorola 6800

Il microprocessore 6800 fu il primo concorrente diretto dell'8080 ed è largamente utilizzato. Tuttavia differisce dallo 8080 e dai suoi discendenti in numerose ed importanti caratteristiche. Il 6800 lavora con una singola alimentazione (+5V) e non richiede un circuito di controllo di sistema. Tuttavia, richiede un clock a due fasi, non compatibile TTL, e perciò ha bisogno di uno speciale circuito generatore di clock. La sua frequenza massima è 1 MHz (esistono versioni più veloci), ma non è in realtà possibile un confronto diretto con il clock a 3 MHz dell'8085. Mentre l'8085 richiede per ogni ciclo di memoria da tre a sei periodi di clock, il 6800 ne richiede uno solo. Perciò un 6800 a 1 MHz può lavorare di fatto più velocemente di un 8085 a 3 MHz.

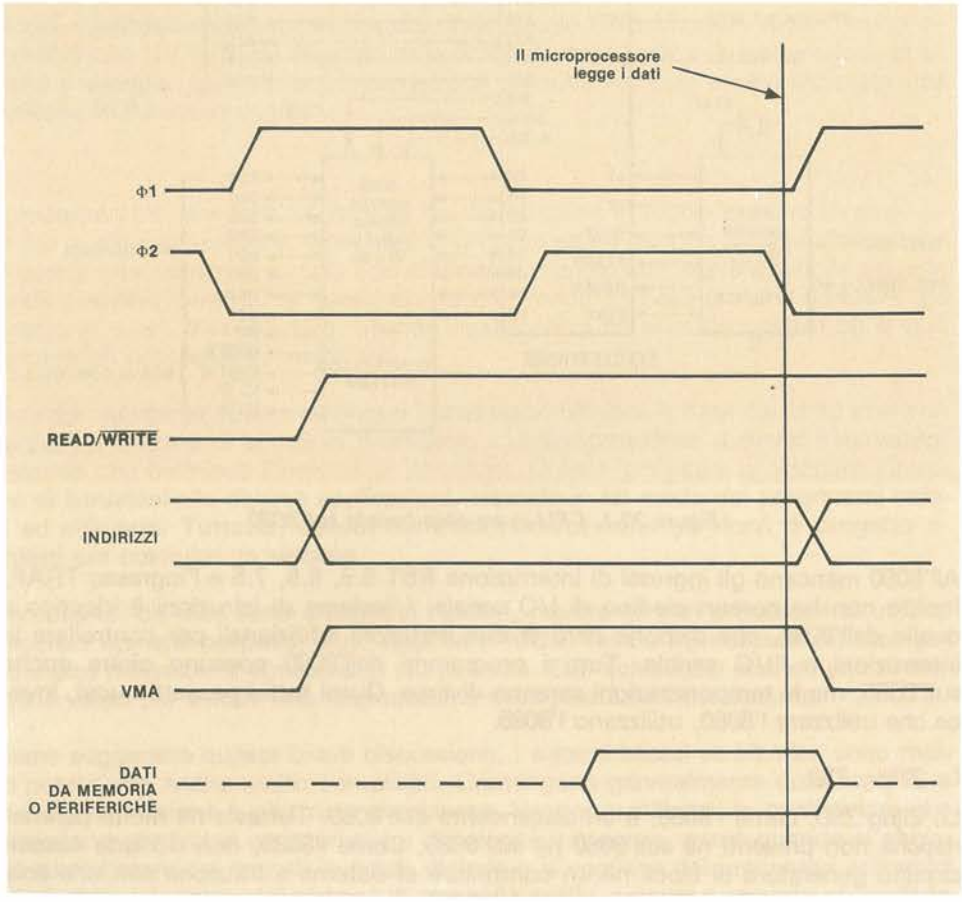


Figura 20-2. Segnali di controllo del 6800: operazione di lettura

I segnali di controllo lettura (read) e scrittura (write) del 6800 sono molto diversi da quelli dell'8085. LA Figura 20-2 riporta le temporizzazioni nel caso di una operazione di lettura. Il clock è costituito da due segnali, $\Phi 1$ e $\Phi 2$, che non si sovrappongono (Nonoverlapping, detto anche clock a due-fasi). Brevemente, dopo il fronte di salita di $\Phi 1$, divengono validi gli indirizzi e il segnale $R/\overline{W}$. Contemporaneamente diviene alto il segnale Valid Memory Address (VMA-Indirizzo di memoria valido), ad indicare che il bus degli indirizzi contiene informazioni valide. $\Phi 2$ segnala al dispositivo indirizzato di porre i dati sul bus, ed infine, al fronte di discesa di $\Phi 2$, questi dati vengono memorizzati nel microprocessore.

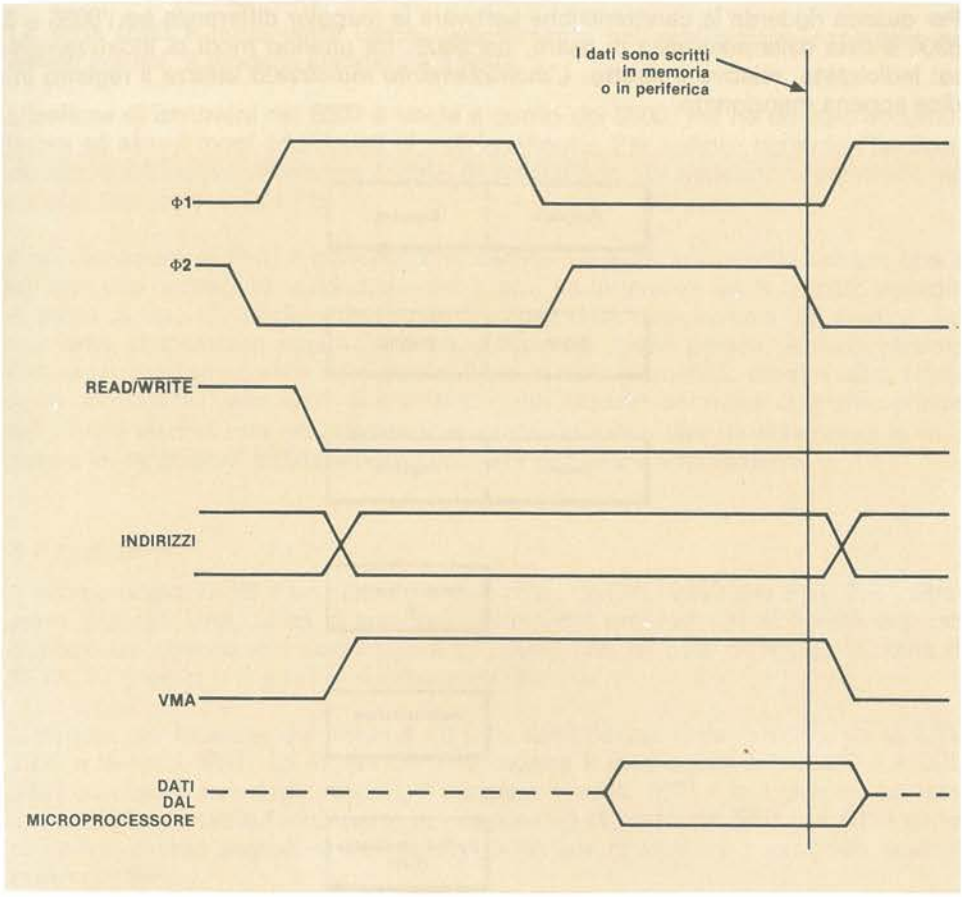


Figura 20-3. 6800: operazione di scrittura

Notate che il significato del segnale $R/\overline{W}$ è diverso da quello dei segnali $\overline{WRITE}$ e $\overline{READ}$ nell'8085. Il segnale $R/\overline{W}$ non indica esattamente l'istante preciso in cui avviene il trasferimento, ma specifica solamente la direzione dello stesso. La temporizzazione per il trasferimento stesso è fornita dal clock $\Phi 2$. Questa temporizzazione è quindi completamente diversa da quella che si ha con l'8085, in cui i segnali $\overline{READ}$ e $\overline{WRITE}$ indicano sia la direzione che i tempi, e dove la frequenza del clock è più elevata di quella del trasferimento.

La Figura 20-3 mostra il diagramma dei tempi del 6800 in caso di scrittura. Tale diagramma è identico a quello che descrive un'operazione di lettura, solo che il segnale $R/\overline{W}$ è basso invece che alto. Il microprocessore mette il dato sul bus quando $\Phi 2$ è alto e, al fronte di discesa di $\Phi 2$, il dispositivo indirizzato memorizza questo dato.

Il software del 6800

L'insieme delle istruzioni del 6800 differisce per numerosi aspetti da quello dell'8085. In Figura 20-4 sono mostrati i registri del 6800. Analogamente all'8085,

possiede un puntatore di stack ed un program counter. È anche dotato di un registro a sedici bit, detto registro indice, e di due accumulatori di otto bit. Non ci sono registri di uso generale (a parte gli accumulatori).

A causa della mancanza di registri di uso generale, per memorizzare temporaneamente i dati, si deve utilizzare quasi sempre la RAM esterna. Questo semplifica l'insieme di istruzioni, perché per ciascuna istruzione non sono necessarie le variazioni che fanno riferimento ai diversi registri. Il secondo accumulatore può essere usato per i risultati o per gli operandi.

Per quanto riguarda le caratteristiche software la maggior differenza tra l'8085 e il 6800 è data dalla possibilità di usare, nel 6800, tre ulteriori modi di indirizzamento: indicizzato, relativo e diretto. L'*indirizzamento indicizzato* utilizza il registro indice appena menzionato.

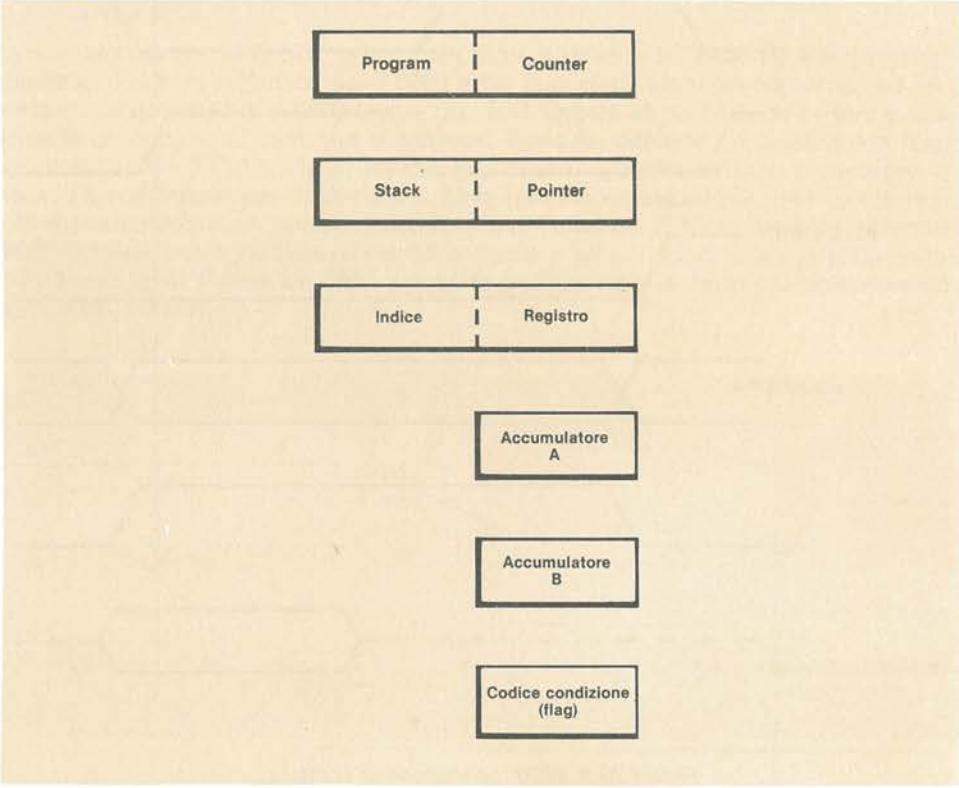


Figura 20-4. Registri della CPU 6800

In questo modo, l'indirizzo effettivo (l'indirizzo realmente inviato alla memoria) è la somma dello spostamento (Offset) indicato all'interno dell'istruzione con il contenuto del registro indice. Come gli altri registri, il registro indice può essere incrementato, decrementato o caricato con un valore voluto. Questa tecnica di indirizzamento è utile per gestire blocchi di dati e per eseguire operazioni che utilizzino tabelle.

L'*indirizzamento relativo* vi permette di specificare un indirizzo relativo all'istruzione corrente. Per esempio, potete scrivere un'istruzione che fa saltare (Jump) avanti di tre indirizzi il processore o che lo fa saltare indietro di sette indirizzi. L'8085, invece, vi richiede sempre di specificare l'indirizzo reale (*indirizzamento assoluto*).

Il modo di *indirizzamento diretto* vi permette di specificare solamente il byte meno significativo dell'indirizzo, assumendo l'altro byte a zero. Questo abbrevia di un byte tutte le istruzioni che operano nelle prime 256 locazioni di memoria.

Il 6802 è una nuova versione del 6800, che contiene, all'interno del circuito integrato della CPU, 128 byte di RAM ed un generatore di clock. L'insieme di istruzioni è identico a quello del 6800. Esistono poi molte altre versioni del 6800.

Il 6800 è stato descritto molto brevemente, ma questa discussione dovrebbe avervi dato un'idea delle differenze tra 6800 e 8085. Informazioni complete si possono ottenere dalla documentazione Motorola.

Il MOS Technology 6500

La serie 6500 è largamente usata, specialmente nell'applicazione consumer ad elevati volumi. Il modello standard è il 6502.

L'insieme di istruzioni del 6502 è simile a quello del 6800, ma ha un solo accumulatore ed alcuni modi addizionali di indirizzamento. Per quanto riguarda l'hardware, una significativa differenza è data dall'inclusione del generatore del clock nel circuito integrato della CPU.

Il microprocessore 6502 è disponibile in diverse versioni a 28 piedini (invece che a 40) con una conseguente riduzione del costo. La differenza tra le quattro versioni è legata al tipo di funzioni soppresse a causa della diminuzione dei piedini, dal momento che devono essere comunque eliminati dodici piedini. Alcune versioni eliminano le interruzioni e solo poche linee di indirizzamento, mentre altre eliminano soprattutto delle linee di indirizzamento. Questo permette di implementare dei piccoli sistemi con un microprocessore economico che ha solamente le funzioni e la capacità di indirizzamento richieste per quella applicazione.

Il Fairchild F8

Il microprocessore F8 è un sistema a due chip. La CPU 3850 e la PSU 3851 (Program Storage Unit, unità di memorizzazione del programma) si combinano per formare un sistema microcalcolatore completo con 64 byte di RAM, 1K byte di ROM, 32 linee di I/O e un temporizzatore (Timer).

L'aspetto più inusuale del sistema F8 è la suddivisione delle funzioni tra la CPU 3850 e la PSU 3851. La Figura 20-5 ne mostra il diagramma a blocchi. La CPU 3850 non ha un bus degli indirizzi, il program counter (PC) e la logica di indirizzamento della memoria fanno parte del dispositivo di memoria 3851. La CPU genera invece cinque segnali di controllo speciali per controllare il program counter posto nel 3851.

La CPU è in grado di caricare un indirizzo nel PC inviando l'indirizzo voluto, in due parti, sul bus dei dati ed istruendo la PSU in modo tale che quel dato venga caricato nel PC. Può anche ordinare alla PSU di incrementare il PC. Se nel sistema esiste più di una PSU, ognuna mantiene il proprio PC. Tutti i PC sono sincronizzati dalla CPU tramite i segnali di controllo. L'eliminazione del bus indirizzi rende disponibili, nella CPU e nella PSU, sedici piedini, che vengono utilizzati per fornire 32 bit di I/O.

Ci sono un certo numero di altri dispositivi nella famiglia F8. Il 3856 è una versione a 2K byte del 3851. Ci sono anche dei circuiti speciali di interfaccia che permettono di utilizzare con il 3850 memorie dinamiche o statiche standard. Il 3870 è la versione su un solo chip dell'F8 e congloba le funzioni del 3850 e 3856, fornendo su di un solo circuito integrato un processore con 2K byte di ROM e 64 byte di RAM. Il 3871 è un processore su un solo chip con 4K byte di ROM e 132 byte di RAM.

L'F8 è largamente utilizzato in applicazioni di controllo, giochi, strumenti ed elettrodomestici. La versione su un solo chip fornisce un sistema molto potente su di un solo circuito integrato.

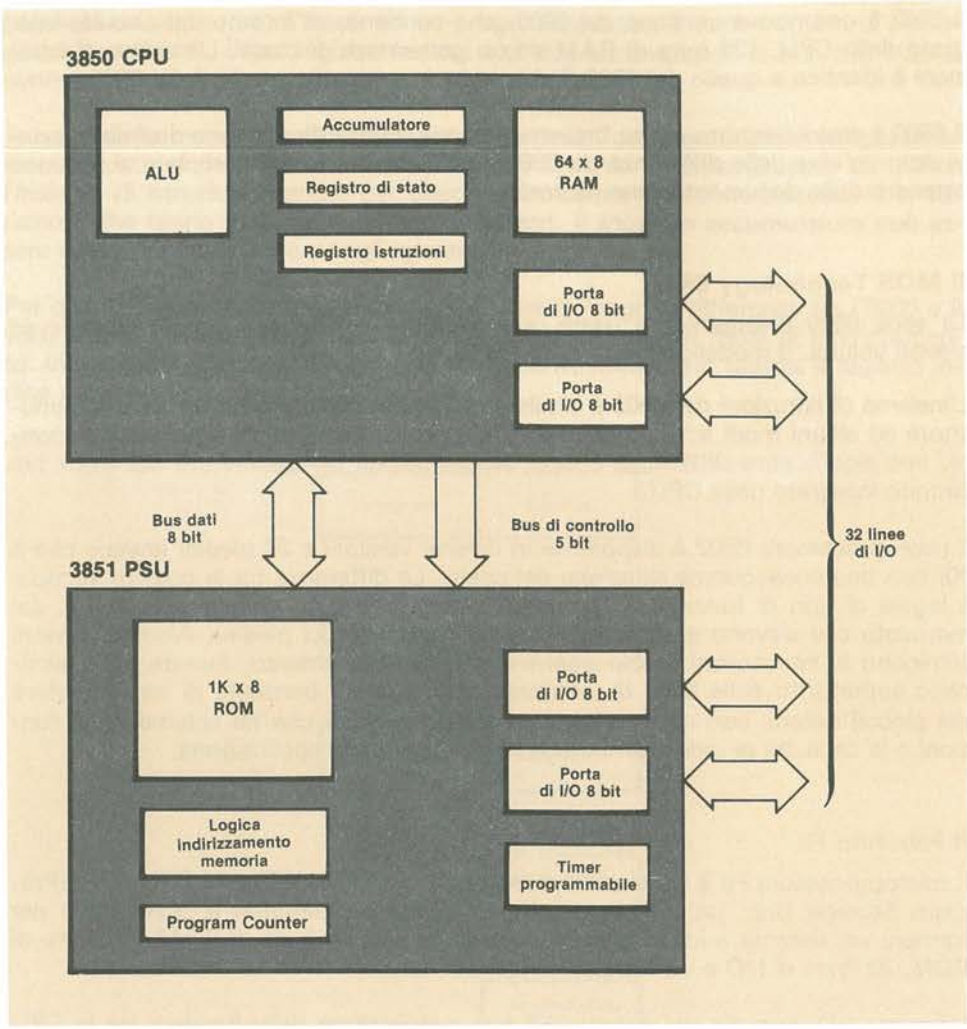


Figura 20-5. Schema a blocchi semplificato del sistema F8

Il Texas Instruments TMS 1000

Il TMS 1000 è in realtà una famiglia di microcalcolatori a quattro bit su un solo chip. Ogni dispositivo contiene su un solo chip, CPU, ROM, RAM, e I/O e per grosse quantità è un processore molto economico. Esistono circa 35 tipi disponibili, con diverse quantità di ROM, RAM e I/O. La ROM può essere di 1 o 2K byte, e la RAM di 64 o 128 parole di quattro bit. Sono disponibili fino a sedici linee di I/O. Questi dispositivi sono largamente utilizzati in applicazioni poco sofisticate ed a basso costo, che prevedono un elevato numero di prodotti.

Sono attualmente disponibili numerosi e diversi tipi di microprocessori. Quelli ad otto bit per applicazioni generali, come l'8085 e il 6800, permettono una notevole flessibilità con una limitata complessità. Per piccoli sistemi, i microcalcolatori su un solo chip come il 3870 o il TMS 1000 offrono una soluzione estremamente semplice ed economica. Per quanto riguarda la fascia alta, ove si hanno applicazioni più esigenti, sono utilizzati i microprocessori a sedici bit che hanno prestazioni analoghe a quelle dei minicalcolatori. Per quelle applicazioni dove è richiesta una elevata velocità sono utilizzati i bit-slice.

Tutti i microprocessori sono riconducibili alla stessa struttura base: una CPU che preleva ed esegue le istruzioni, della memoria per memorizzare istruzioni e dati e delle porte di I/O per comunicare con altri dispositivi. I vari microprocessori hanno diversi insiemi di istruzioni ed utilizzano registri e modi di indirizzamento diversi. Molte applicazioni possono essere realizzate utilizzando uno qualsiasi dei microprocessori disponibili, sebbene la quantità di ROM e RAM richiesta possa variare in modo considerevole. Sono sempre presenti i bus dei dati e degli indirizzi, anche se qualche volta sono nascosti dentro un circuito integrato. I segnali di controllo possono non essere sempre identici, ma hanno sempre le stesse funzioni: indicare alla memoria o alle porte di I/O quando la CPU deve leggere o scrivere un dato e permettere ad altri dispositivi di interrompere o fermare (halt) il microprocessore.

Quando dovete studiare un nuovo microprocessore, fate attenzione alle caratteristiche appena discusse. Da quanti bit è formato il bus dei dati? Quali sono i registri nella CPU? Quali modi di indirizzamento sono disponibili? I dati e gli indirizzi (o i protocolli) sono sullo stesso bus (multiplex)? Quanti circuiti integrati occorrono per realizzare un sistema completo? Quali tipi di chip di supporto sono disponibili?

La varietà e la sofisticazione dei microprocessori sta aumentando rapidamente. Quando avrete compreso i concetti di base e saprete qualcosa sui microprocessori già esistenti, sarete in grado di analizzare i nuovi dispositivi e potrete tenervi al passo in un campo, come questo, in rapida crescita.

Lezione 20

1. L'8085 e lo Z80 sono un'evoluzione dello _____.

2. Il 6502 e il 6802 sono un'evoluzione del _____.

3. I processori a quattro bit sono usati (al posto dei processori a otto bit), in quanto:
   a. sono più veloci.
   b. possono gestire più memoria.
   c. sono più facili da programmare.
   d. sono più economici.

4. I processori a sedici bit sono usati (al posto dei processori a otto bit), in quanto:
   a. sono più economici.
   b. assorbono meno potenza.
   c. possono operare su un campo di numeri più esteso.
   d. possono eseguire più rapidamente i calcoli.

5. I microcalcolatori su un solo chip contengono generalmente:
   a. ROM, RAM e porte I/O.
   b. ROM, RAM ma richiedono porte I/O.
   c. solamente delle porte I/O.
   d. RAM e I/O ma non ROM.

6. I microprocessori bit-slice sono:
   a. facili da usare.
   b. economici.
   c. molto flessibili e potenti.
   d. molto simili ai processori normali.

7. Nel caso di diversi tipi di microprocessori, i segnali di controllo della memoria:
   a. sono tutti uguali.
   b. variano nei dettagli, ma realizzano sempre la stessa funzione.
   c. possono realizzare una larga varietà di funzioni.
   d. sono sempre identici ai segnali di controllo dell'8085.

APPENDICI

APPENDICE A

Soluzioni dei problemi

Questa appendice vi dà le soluzioni di tutti i problemi presentati nel corso del testo. Sono date quattro tipi di soluzioni:

- Risposte alle domande
- Soluzioni degli esercizi di programmazione
- Codice macchina dei programmi presentati negli esperimenti
- Soluzioni ai problemi di ricerca guasti.

Per le soluzioni di problemi di ricerca guasti, date a partire da pagina 353, si suppone che nel μLab sia presente un solo guasto per volta. Facendo uso del diagramma di flusso generale per la ricerca guasti nel μLab (Figura 19-1), vengono sottolineati i successivi passi che portano a localizzare i guasti. Sul diagramma di flusso viene mostrato il percorso da seguire nella ricerca per ognuno dei guasti e ad esso potete fare riferimento prima di leggere la soluzione effettiva del guasto. Nel corso di queste soluzioni, con riferimento ad una ben precisa tabella di verifica delle firme, basata su di una modesta conoscenza del circuito, sono spesso richiamati circuiti integrati o aree di memoria particolari. Queste "indicazioni guidate" sono presentate perché evitiate di sprecar tempo verificando tutte le firme di una tabella. Tale procedimento non è tuttavia necessario al fine di trovare le firme errate.

RISPOSTE ALLE DOMANDE

Lezione 1

1. c
2. b
3. degli indirizzi, dei dati e di controllo
4. byte
5. b
6. d
7. a
8. d
9. a

Lezione 2

1. 1 o 0
2. 172
3. 254
4. AC
5. 1011 1001
6. FF (hex) = 377 (ottale)
7. c

Lezione 3

1. c
2. di assemblaggio, macchina
3. b
4. d
5. a

Lezione 4

1. FETCH ADRS
2. DECR oppure FETCH ADRS 0 8 0 2
3. a
4. indirizzi
5. b

Lezione 5

1. a
2. c
3. registri (accumulatori)
4. CALL
5. RET
6. b
7. a

Lezione 6

1. fetch
2. c
3. hardware
4. c
5. b

Lezione 7

1. dei dati, degli indirizzi di controllo
2. b
3. a
4. b
5. c
6. c
7. d
8. $\overline{\text{WRITE}}$, $\overline{\text{READ}}$

Lezione 8

1. a
2. d
3. c
4. b
5. a
6. c

Lezione 9

1. c
2. c
3. a
4. d
5. c
6. c
7. d
8. d

Lezione 10

1. b
2. a
3. d
4. a
5. d
6. b
7. c
8. a

Lezione 11

1. b
2. d
3. STORE/INCR tre volte
4. a

Lezione 12

1. b
2. a
3. c
4. 0930
5. c

Lezione 13

1. il problema
2. c
3. a
4. d

Lezione 14

1. c
2. b
3. b
4. c

Lezione 15

1. d
2. b
3. virgola mobile
4. a
5. b

Lezione 16

1. b
2. c
3. d
4. d
5. a
6. b

Lezione 17

1. d
2. d
3. d
4. c
5. b
6. d
7. a
8. b

Lezione 18

1. a
2. d
3. d
4. d
5. a
6. c

Lezione 19

1. d
2. d
3. b
4. b
5. d

Lezione 20

1. 8080
2. 6800
3. d
4. d
5. a
6. c
7. b

SOLUZIONI AGLI ESERCIZI DI PROGRAMMAZIONE

Nota: Queste soluzioni riguardano i soli esercizi di programmazione.
Per i programmi utilizzati negli esperimenti si passi alla sezione seguente.

Esercizio di programmazione 12-1: Mascheratura

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	3A	START:	LDA 2000	; Leggi porta d'ingresso nell'accumulatore
0801	00			
0802	20			
0803	06		MVI B,08	; Carica B con il valore della maschera (00001000 binario)
0804	08			
0805	A0		ANA B	; Metti a zero tutti i bit escluso il bit 3
0806	CA		JZ OFF	; Verficare se l'accumulatore è = 0
0807	11			
0808	08			
0809	3E	ON:	MVI A,0	; Accendi i LED
080A	00			
080B	32		STA 3000	
080C	00			
080D	30			
080E	C3		JMP START	
080F	00			
0810	08			
0811	3E	OFF:	MVI A,FF	; Spegni i LED
0812	FF			
0813	32		STA 3000	
0814	00			
0815	30			
0816	C3		JMP START	
0817	00			
0818	08			

Esercizio di programmazione 12-2: Rotazione

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	3E		MVI A,FE	; Carica l'accumulatore con 1111 1110
0801	FE			
0802	32	LOOP:	STA 3000	; Scrivi nella porta d'uscita
0803	00			
0804	30			
0805	0F		RRC	; Ruota il dato
0806	C3		JMP LOOP	; Ripeti
0807	02			
0808	08			

Utilizzo dei breakpoint nell'esercizio 12-2

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	3E		MVI A,FE	; Carica l'accumulatore con 1111 1110
0801	FE			
0802	32	LOOP:	STA 3000	; Scrivi nella porta d'uscita
0803	00			
0804	30			
0805	CF		RST 1	; Ruota il dato
0806	0F		RRC	; Breakpoint
0807	C3		JMP LOOP	; Ripeti
0808	02			
0809	08			

Esercizio di programmazione 13-1a: Controllore di semafori con durata di giallo e di verde variabile in modo indipendente

Programma principale

Indirizzo	Contenuto	Label	Istruzione	Commenti
080C	2E	TRAF:	MVI L,1	; Stabilisci la durata del giallo per A
080D	01			
080E	16		MVI D,6	; Stabilisci la durata del verde per A
080F	06			
0810	1E		MVI E,0	; Sequenza per il semaforo A
0811	00			
0812	CD		CALL SEQ	
0813	30			
0814	08			
0815	2E		MVI L,3	; Stabilisci la durata del giallo per A
0816	03			
0817	16		MVI D,10	; Stabilisci la durata del verde per B
0818	10			
0819	1E		MVI E,1	; Sequenza per il semaforo B
081A	01			
081B	CD		CALL SEQ	
081C	30			
081D	08			
081E	C3		JMP TRAF	
081F	0C			
0820	08			

Routine sequenza

0830	26	SEQ:	MVI H,7D	; Accendi il verde
0831	7D			
0832	CD		CALL CHNG	
0833	55			
0834	08			
0835	00		NOP	; Attendi la durata del verde
0836	00		NOP	
0837	CD		CALL DELAY	; Attenti la durata del verde
0838	70			
0839	08			
083A	26		MVI H,7B	; Accendi il giallo
083B	7B			
083C	CD		CALL CHNG	
083D	55			
083E	08			
083F	55		MOV D,L	; Attendi la durata del giallo
0840	00			
0841	CD		CALL DELAY	
0842	70			
0843	08			
0844	C9		RET	

(Le Routine CHNG e DELAY sono identiche a quelle della Tabella 13-8)

Esercizio di programmazione 13-1b: Controllore di semafori con un periodo di rosso contemporaneo per entrambi i semafori

Programma principale

Indirizzo	Contenuto	Label	Istruzione	Commenti
080E	16	TRAF:	MVI D,6	; Stabilisci la durata del verde per A
080F	06			
0810	1E		MVI E,0	; Sequenza per il semaforo A
0811	00			
0812	CD		CALL SEQ	
0813	30			
0814	08			
0815	00		NOP	
0816	00		NOP	
0817	16		MVI D,10	; Stabilisci la durata del verde per B
0818	10			
0819	1E		MVI E,1	; Sequenza per il semaforo B
081A	01			
081B	CD		CALL SEQ	
081C	30			
081D	08			
081E	C3		JMP TRAF	
081F	0E			
0820	08			

Routine sequenza

0830	26	SEQ:	MVI H,7D	; Accendi il verde
0831	7D			
0832	CD		CALL CHNG	
0833	55			
0834	08			
0835	00		NOP	; Attendi la durata del verde
0836	00		NOP	
0837	CD		CALL DELAY	
0838	70			
0839	08			
083A	26		MVI H,7B	; Accendi il giallo
083B	7B			
083C	CD		CALL CHNG	
083D	55			
083E	08			
083F	16		MVI D,2	; Attendi la durata del giallo
0840	02			
0841	CD		CALL DELAY	
0842	70			
0843	08			
0844	26		MVI H,77	; Accendi il rosso per entrambi i semafori
0845	77			
0846	CD		CALL CHNG	
0847	55			
0848	08			
0849	16		MVI D,1	; Attendi la durata del rosso doppio
084A	01			
084B	CD		CALL DELAY	
084C	70			
084D	08			
084E	C9		RET	

(Le routine CHNG e DELAY sono identiche a quelle della Tabella 13-8)

Esercizio di programmazione 13-1c: Controllore di semafori con frecce di svolta a sinistra

Programma principale

Indirizzo	Contenuto	Label	Istruzione	Commenti
080E	2E	TRAF:	MVI L,6	; Stabilisci la durata del verde per A
080F	06			
0810	1E		MVI E,0	; Sequenza per il semaforo A
0811	00			
0812	CD		CALL SEQ	
0813	26			
0814	08			
0815	00		NOP	
0816	00		NOP	
0817	2E		MVI L,10	; Stabilisci la durata del verde per B
0818	10			
0819	1E		MVI E,1	; Sequenza per il semaforo B
081A	01			
081B	CD		CALL SEQ	
081C	26			
081D	08			
081E	C3		JMP TRAF	
081F	0E			
0820	08			

Routine di sequenza

0826	26	SEQ:	MVI H,7E	; Accendi la freccia di svolta a sinistra
0827	7E			
0828	CD		CALL CHNG	
0829	55			
082A	08			
082B	16		MVI D,3	; Attendi per la durata della freccia
082C	03			
082D	CD		CALL DELAY	
082E	70			
082F	08			
0830	26		MVI H,7D	; Accendi il verde
0831	7D			
0832	CD		CALL CHNG	
0833	55			
0834	08			
0835	55		MOV D,L	; Attendi per la durata del verde
0836	00		NOP	
0837	CD		CALL DELAY	
0838	70			
0839	08			
083A	26		MVI H,7B	; Accendi il giallo
083B	7B			
083C	CD		CALL CHNG	
083D	55			
083E	08			
083F	16		MVI D,2	; Attendi per la durata del giallo
0840	02			
0841	CD		CALL DELAY	
0842	70			
0843	08			
0844	C9		RET	

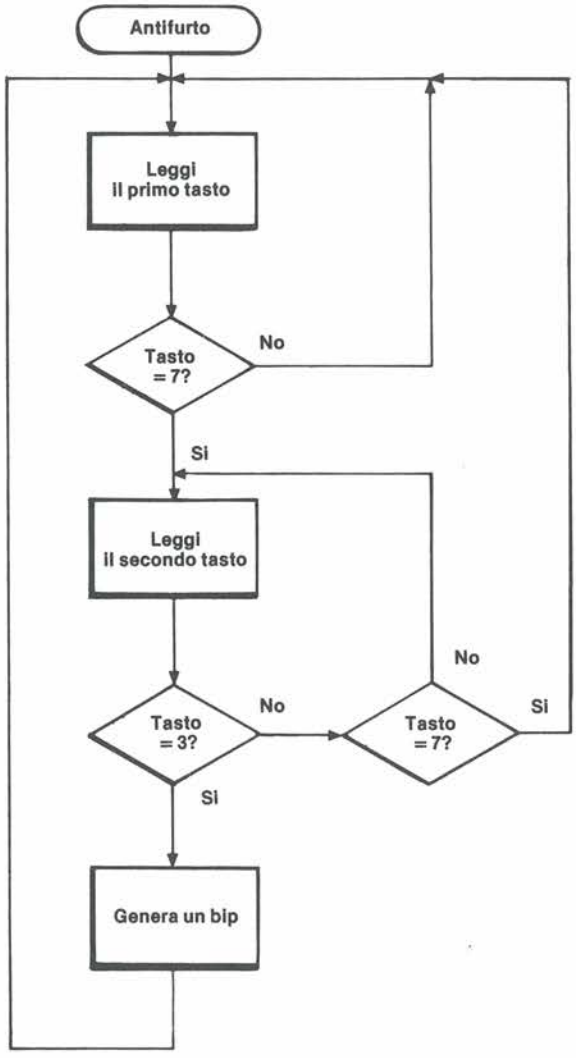
(Le routine CHNG e DELAY sono identiche a quelle della Tabella 13-8)

SOLUZIONI AGLI ESERCIZI DI PROGRAMMAZIONE

(continuazione)

Esercizio di programmazione 14-1: Antifurto Elettronico

Diagramma di flusso



Programma per l'esercizio 14-1

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	CD	FIRST:	CALL KIND	; Leggi il primo tasto
0801	4B			
0802	01			
0803	FE		CPI 07	; Tasto = 7?
0804	07			
0805	C2		JNZ FIRST	; Se no, rileggi
0806	00			
0807	08			
0808	CD	SECOND:	CALL KIND	; Leggi il secondo tasto
0809	4B			
080A	01			
080B	FE		CPI 03	; Tasto = 3?
080C	03			
080D	CA		JZ OPEN	; Se sì, genera un bip
080E	18			
080F	08			
0810	FE		CPI 07	; Tasto = 7?
0811	07			
0812	CA		JZ SECOND	; Se sì, leggi il secondo tasto
0813	08			
0814	08			
0815	C3		JMP FIRST	; Se no, leggi il primo tasto
0816	00			
0817	08			
0818	CD	OPEN:	CALL BEEP	; Combinazione corretta, genera un bip
0819	10			
081A	00			
081B	C3		JMP FIRST	; Ripeti.
081C	00			
081D	08			

Esercizio di programmazione 14-2: Uso della tastiera e del display

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	CD	LOOP:	CALL KIND	; Leggi il tasto
0801	4B			
0802	01			
0803	32		STA 0B00	; Memorizza il tasto in RAM
0804	00			
0805	0B			
0806	11		LXI D,MESS	; Prepara l'indirizzo del messaggio
0807	00			
0808	0B			
0809	CD		CALL SDM	; Trasferisci il messaggio
080A	18			
080B	00			
080C	CD		CALL DCD	; Visualizza il messaggio
080D	E9			
080E	01			
080F	C3		JMP LOOP	; Ripeti
0810	00			
0811	08			
0B00	00	MESS:	(Riservato per dato del tasto)	
0B01	10			
0B02	10			
0B03	10		Spazi	
0B04	10			
0B05	10			

PROGRAMMI PER ESPERIMENTI

Esperimento 12-1: Programma per spiegare le istruzioni logiche

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	3A	START:	LDA 2000	; Leggi la porta d'ingresso
0801	00			
0802	20			
0803	06		MVI B,3C	; Carica il registro B con 0011 1100
0804	3C			
0805	A0		ANA B	; AND dell'accumulatore con B
0806	32		STA 3000	; Invia il risultato alla porta d'uscita
0807	00			
0808	30			; Ripeti
0809	C3		JMP START	
080A	00			
080B	08			

Esperimento 12-2: Esempio di programma con istruzioni automatiche

Indirizzo	Contenuto	Istruzione	Commenti
0800	90	SUB B	; Sottrai B all'accumulatore
0801	81	ADD C	; Somma C all'accumulatore

Esperimento 14-1: Programma per leggere la tastiera e generare un bip se viene premuto il tasto "7"

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	CD	READ:	CALL KIND	; Lettura tasto
0801	4B			
0802	01			
0803	FE		CPI 07	; Confrontare il codice tasto
0804	07			
0805	C2		JNZ READ	
0806	00			
0807	08			
0808	CD		CALL BEEP	; Genera un bip se il tasto è "7".
0809	10			
080A	00			
080B	C3		JMP READ	
080C	00			
080D	08			

Per rilevare il tasto "E", modifica la locazione 0804 in 0E

Esperimento 14-2: Programma che verifica quando è premuto il tasto 2

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	3E		MVI A,F7	; Definisci sulla porta di scansione il valore 1111 0111
0801	F7			
0802	32		STA 2800	
0803	00			
0804	28			
0805	3A	READ:	LDA 1800	; Leggi le colonne
0806	00			
0807	18			
0808	06		MVI B,07	; Tieni solo i tre LSB e maschera tutti gli altri a zero
0809	07			
080A	A0		ANA B	
080B	FE		CPI 05	; Il dato è 101 (tasto "2")?
080C	05			
080D	C2		JNZ READ	; Se no, continua a leggere
080E	05			
080F	08			
0810	CD		CALL BEEP	; Se sì, genera un bip
0811	10			
0812	00			
0813	C3		JMP READ	; Riprendi la lettura
0814	05			
0815	08			

Per rilevare il tasto "3", modifica la locazione 080C in 3.

Esperimento 14-3: Programma per visualizzare un messaggio

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	11		LXI D,0810	; Prepara l'indirizzo del messaggio
0801	10			
0802	08			
0803	CD		CALL STDM	; Trasferisci il messaggio
0804	18			
0805	00			
0806	CD	LOOP:	CALL DCD	; Visualizza il messaggio
0807	E9			
0808	01			
0809	C3		JMP LOOP	
080A	06			
080B	08			

Esperimento 14-4: Programma per visualizzare un "2"

Indirizzo	Contenuto	Label	Istruzione	Commenti
0800	3E	START:	MVI A,4	; Carica la porta di scansione per selezionare il digit (4 hex = 0000 0100 in binario)
0801	04			
0802	32		STA 2800	
0803	00			
0804	28			
0805	3E		MVI A,A4	; Carica la porta dei segmenti per visualizzare il carattere "2"
0806	A4			
0807	32		STA 3800	
0808	00			
0809	38			
080A	C3		JMP START	
080B	00			
080C	08			

SOLUZIONE DEL GUASTO 1

Problema:

A) Display: Fisso, informazione casuale
B) LED d'uscita: Fissi, informazione casuale
C) Tastiera: Nessuna risposta

Accensione: Scorretta

Luci accese: Sì

Attività del bus: Sì

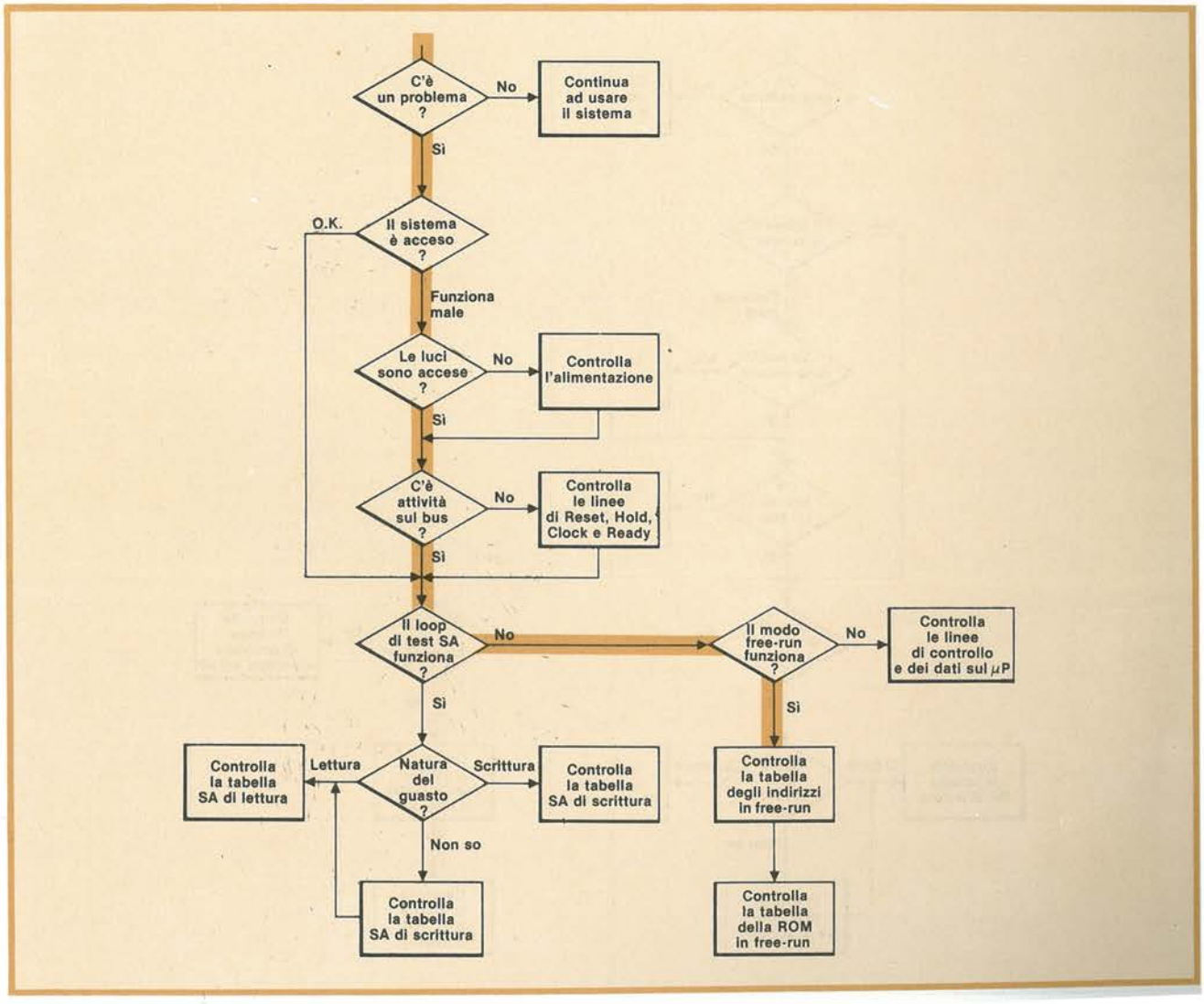
Funziona il loop di test SA: No

Funziona il test free-run = Sì

Tabella C1, A0—A15: Viene trovata una firma scorretta

A) La firma di A11 è scorretta (HPP0)
B) La firma di A11 su IC1-3 (uscita del buffer degli indirizzi) è corretta (1293)
C) Con l'analizzatore di firma seguite il percorso del segnale tra IC1-3 e il punto di prova A11 sul bus.
D) Localizzate il Guasto 1, un cortocircuito tra le linee del bus A10 e A11.

Percorso di ricerca guasti per il Guasto 1



SOLUZIONE DEL GUASTO 2

Problema:

A) Display: Fisso
B) LED d'uscita: Fissi
C) Tastiera: Nessuna risposta

Accensione: Scorretta

Luci accese: Sì

Attività del bus: Sì

Funziona il loop di test SA: No

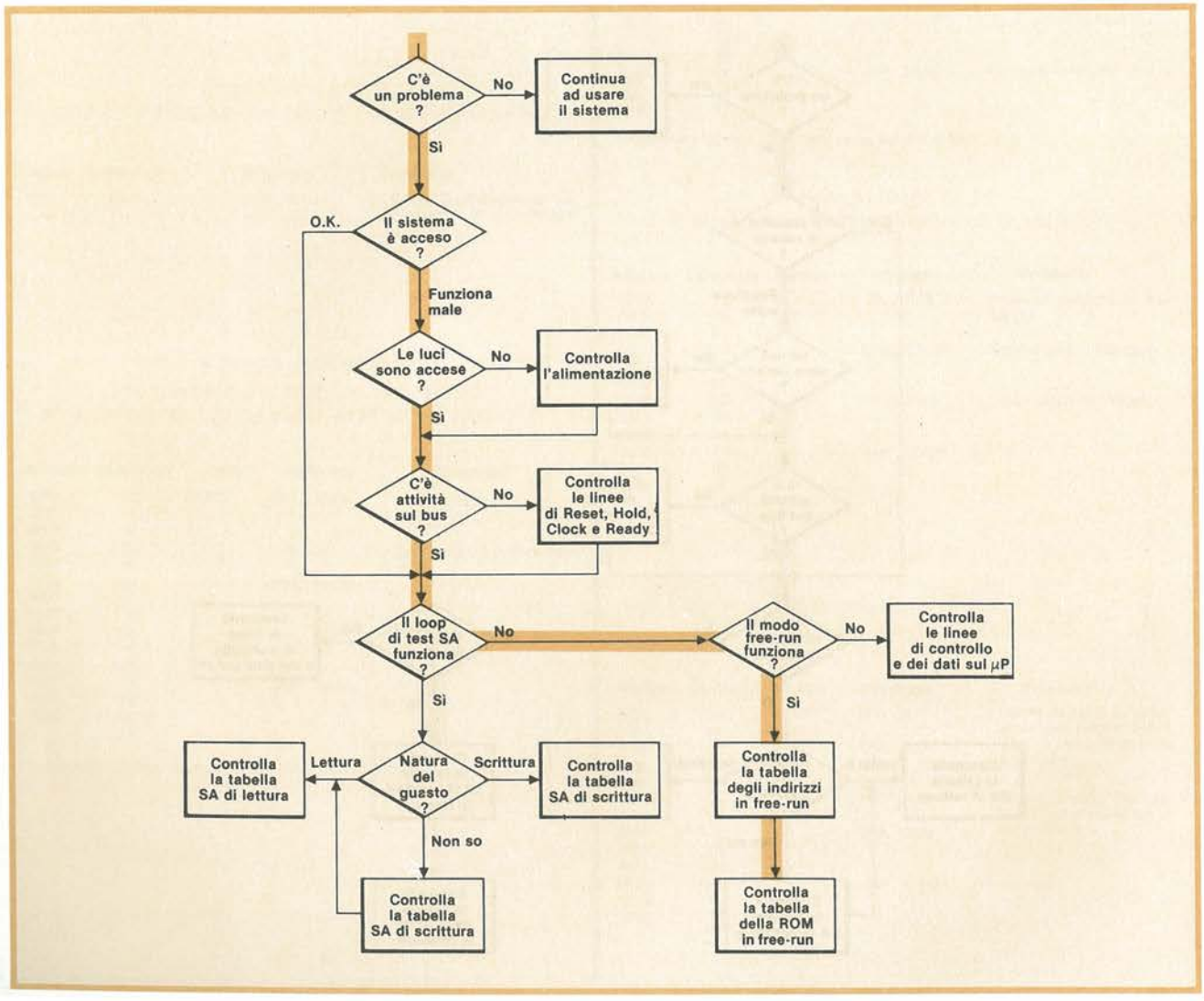
Funziona il test free-run: Sì

Tabella C-1 A0-A15: Le firme di tutti gli indirizzi sono corrette

Tabella C-2: Viene trovata una firma scorretta.

A) La firma di D5 è instabile.
B) La firma di D5 su IC4-15 (uscita della ROM) è corretta (0HF1).
C) Con l'analizzatore di firma seguite il percorso del segnale tra IC4-15 e il punto di prova D5.
D) Localizzate il Guasto 1, una pista coperta sull'uscita di D5 dalla ROM.

Percorso di ricerca guasti per il Guasto 2



SOLUZIONE DEL GUASTO 3

Problema:
- A) Display: Fisso
- B) LED d'uscita: Fissi
- C) Tastiera: Nessuna risposta

Accensione: Scorretta

Luci accese: Sì

Attività del bus: Sì

Funziona il loop di test SA: No

Funziona il test free-run: Sì. Sulle linee A0-A7 non viene rilevata alcuna attività

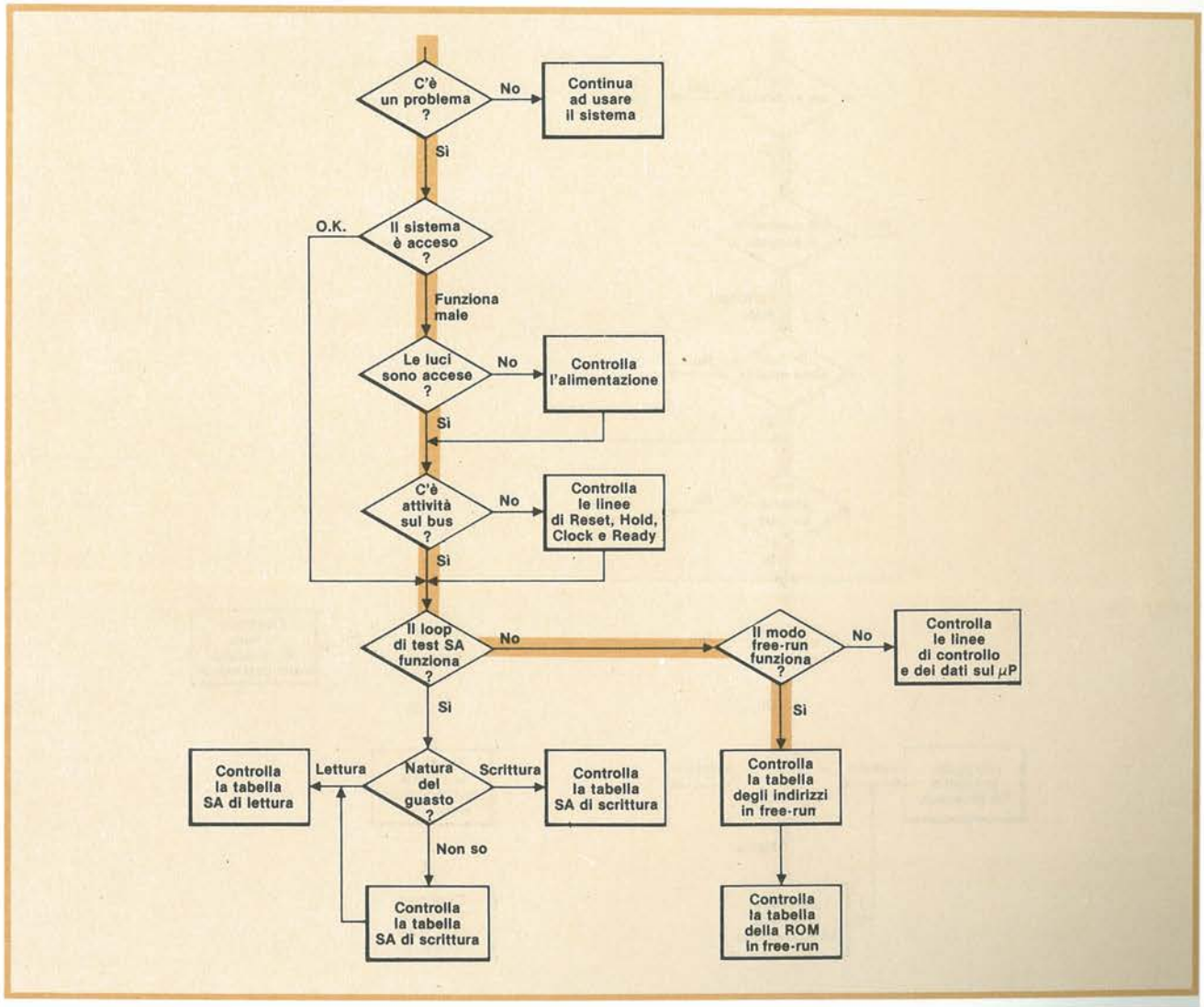
Tabella C-1, A0-A15: Vengono trovate firme scorrette.

- A) Le linee A0-A7 sono bloccate (nessuna attività).
- B) Le linee A0-A7 sono bloccate sulle uscite di IC2.
- C) Le linee AD0-AD7 non sono bloccate agli ingressi di IC2.
- D) IC2 e il segnale $\overline{ALE}$ sono sospetti.

Tabella C-1, IC2: È trovata una firma scorretta per ALE.

- A) $\overline{ALE}$ su IC2-11 è bloccato alto.
- B) $\overline{ALE}$ su IC12-4 è bloccato alto.
- C) $\overline{ALE}$ su IC12-3 ha la firma corretta.
- D) $\overline{ALE}$ è il segnale guasto. Il guasto si trova su IC12-4, IC10-1, IC2-11 o su una pista della scheda.
- E) Fate uso del rivelatore di corrente per controllare che in IC12-4 ci sia corrente. Ponete la punta del rivelatore proprio sopra IC12-4 e impostate un valore medio di sensibilità. La corrente presente su questo piedino si rivela essere assai elevata, dal che si può dedurre che IC8 sta cercando di pilotare il modo. Il guasto deve essere perciò altrove.
- F) Seguite il percorso della corrente lungo la pista che porta a IC10-1 e quindi prosegue. Osservate che la corrente non entra in IC10-1.
- G) Continuate a seguire la corrente verso IC2-11.
- H) La corrente scompare nel momento in cui passate oltre il Guasto 3. Il Guasto 3 mette la linea $\overline{ALE}$ in corso con la massa.

Percorso di ricerca guasti per il Guasto 3



SOLUZIONE DEL GUASTO 4

Problema:

A) Display: segnala un guasto su IC6

Accensione: Scorretta

Luci accese: Sì

Attività del bus: Sì

Funziona il loop di test SA: Sì.

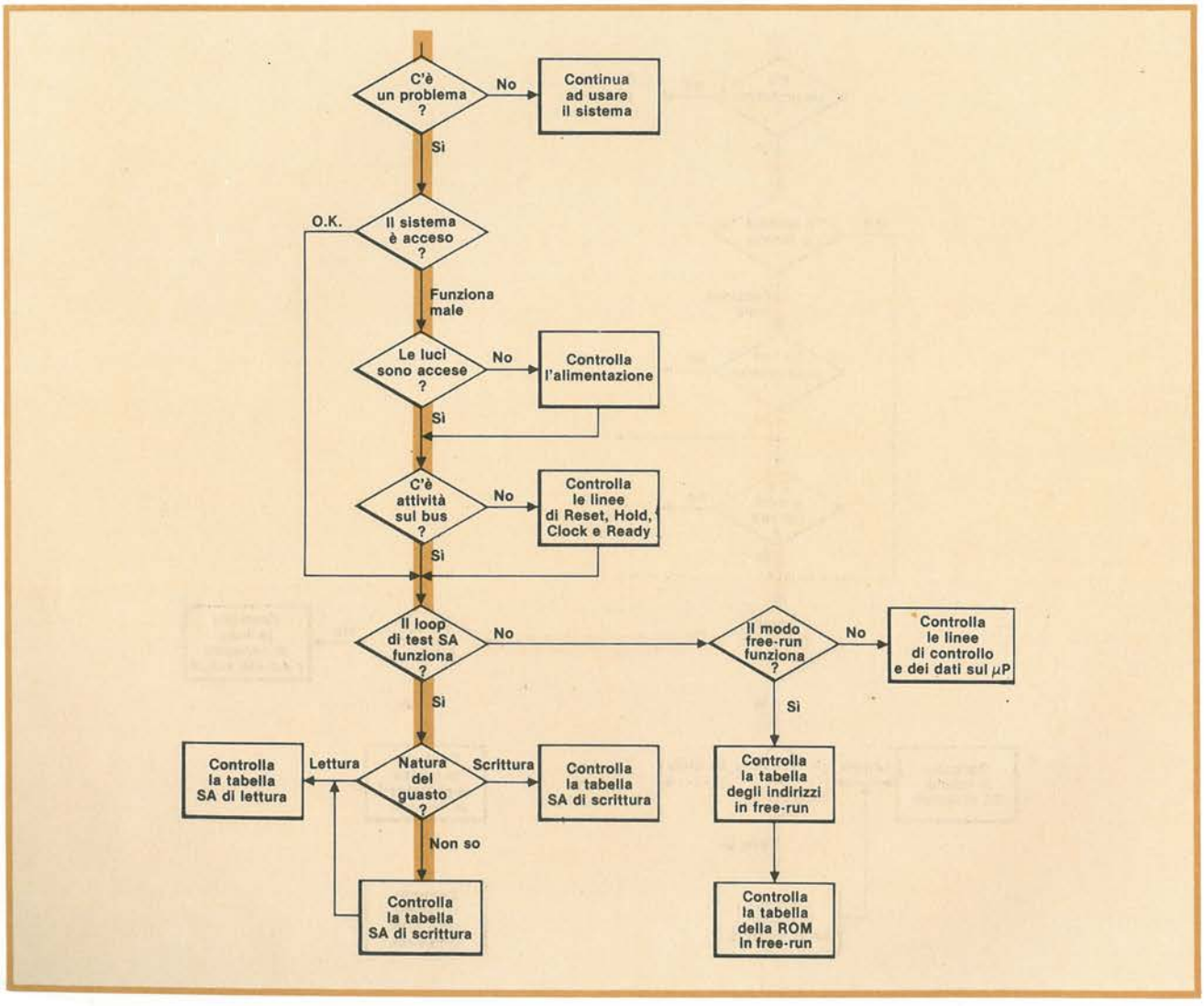
Possono essere guasti un circuito integrato della RAM (IC6) o i segnali di controllo ad esso collegati. Se tali segnali di controllo dovessero risultare corretti, sarebbe allora guasta la RAM IC6. Parte di essa comunque deve essere a posto perché in caso contrario sia la tastiera che il display opererebbero scorrettamente.

Tabella C-3, IC6: Viene trovata una firma scorretta.

A) A7, su IC6-17 risulta essere bloccato alto.
B) A7 sul bus degli indirizzi ha una firma corretta.
C) Facendo uso dell'analizzatore di firma, seguite il percorso del segnale tra IC6-17 e il punto di prova A7.
D) Localizzate il Guasto 4, la linea A7 di indirizzamento della RAM è collegata a Vcc.

Il fatto che un ingresso di una linea d'indirizzo sia guasto può essere dovuto ad una pista aperta o ad un foro non metallizzato, col risultato comunque che il piedino d'indirizzo della RAM è fluttuante. Un piedino d'ingresso della RAM che sia aperto interamente provocherà un analogo malfunzionamento, ma non genererà una firma scorretta sul piedino del dispositivo.

Percorso di ricerca guasti per il Guasto 4



SOLUZIONE DEL GUASTO 5

Problema:

A) Display: Fisso
B) LED d'uscita: Tremolanti
C) Tastiera: Nessuna risposta.

Accensione: Scorretta

Luci accese: Sì

Attività del bus: Sì

Funziona il loop di test SA: No

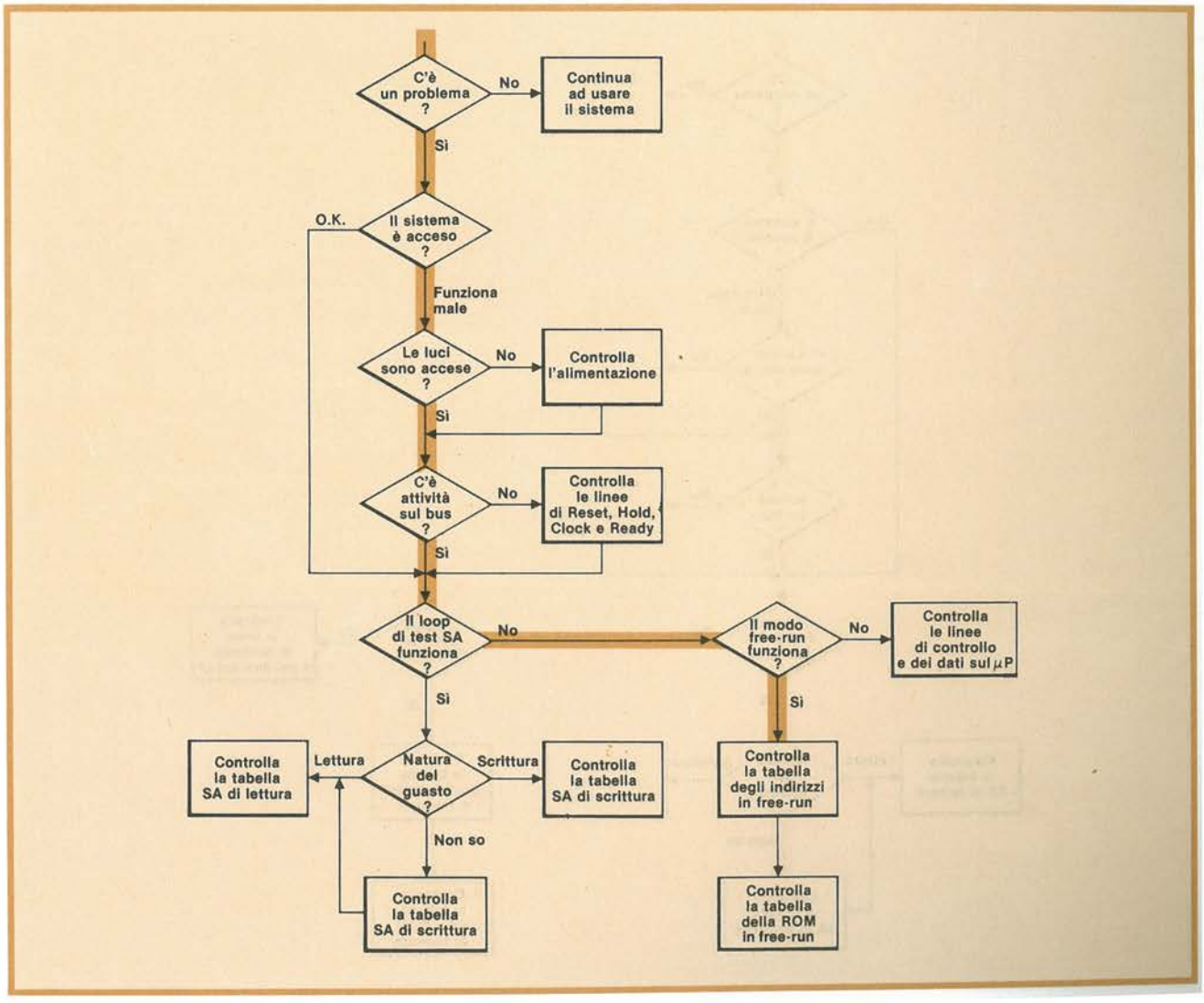
Funziona il test free-run: Sì

Tabella C-1, A0-A15: Tutte le firme sono corrette

Tabella C-1, IC7: Vengono trovate delle firme scorrette.

A) Firme scorrette su IC7, piedini 3, 7, 9, 10, 11, 12, 13, 14, 15.
B) Sarebbe sufficiente che fosse guasto il solo piedino 3 per generare tutte le altre firme scorrette.
C) Sul piedino 3 non è presente alcuna attività e la punta della sonda dell'analizzatore di firma rivela un cattivo livello logico. Si può sospettare che la linea A13 sia aperta.
D) Seguite il percorso del segnale tra IC7-3 e il punto di prova A13.
E) Localizzate il Guasto 5, un collegamento aperto tra A13 e IC7-3.

Percorso di ricerca guasti per il Guasto 5



SOLUZIONE DEL GUASTO 6

Problema:

A) Display: Segnala un errore SP.
B) Non è possibile modificare i dati nella parte protetta della RAM (0800-0AFF). I dati possono essere modificati nella parte non protetta della RAM (0800-OBFF).

Accensione: Scorretta

Luci accese: Sì

Attività del bus: Sì.

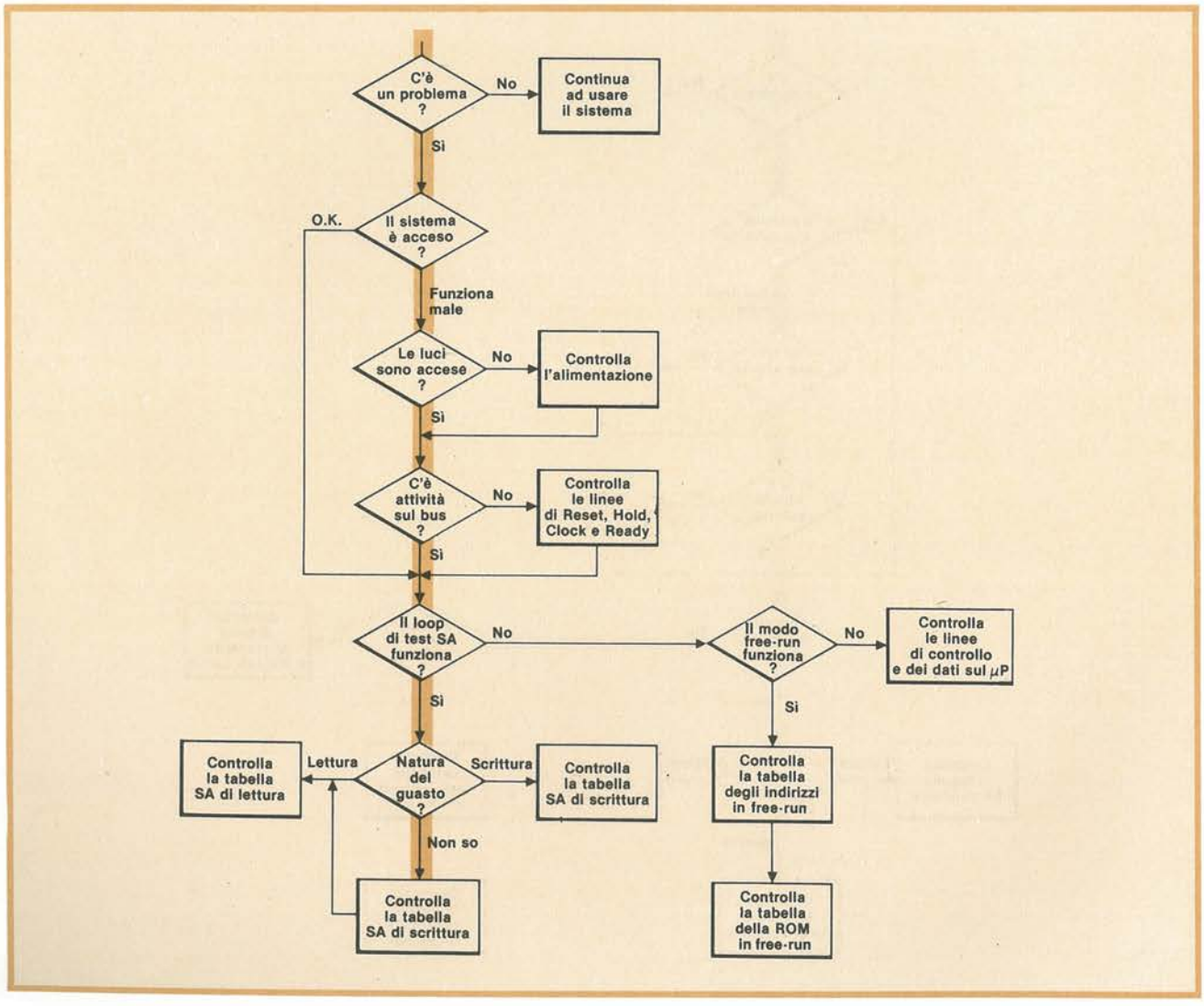
Funziona il loop di test SA: Sì.

La parte protetta della RAM si rivela protetta dalla RAM in qualunque momento. Per prima cosa bisognerà verificare i circuiti che generano i segnali di controllo della scrittura in RAM.

Tabella C-3, IC11, 12, 5, 6: Vengono trovate delle firme scorrette.

A) È possibile seguire una firma scorretta di WR, presente su IC5 o 6, piedino 10 e risalire via via, passando per una serie di firme scorrette su IC11 piedini 3 e 2, IC12 piedini 2 e 1, IC11 piedini 6 e 4, ad una firma corretta su IC8-6.
B) Poiché IC11-4 e IC8-6 dovrebbero essere collegati (ed avere quindi la stessa firma), si può supporre che tra i due punti ci sia un'interruzione.
C) Seguite il percorso del segnale tra IC11-4 e IC8-6.
D) Localizzate il Guasto 6, un corto verso massa su IC11-4.

Percorso di ricerca guasti per il Guasto 6



SOLUZIONE DEL GUASTO 7

Problema:

A) Display: Fisso
B) LED d'uscita: Fissi
C) Tastiera: Nessuna risposta

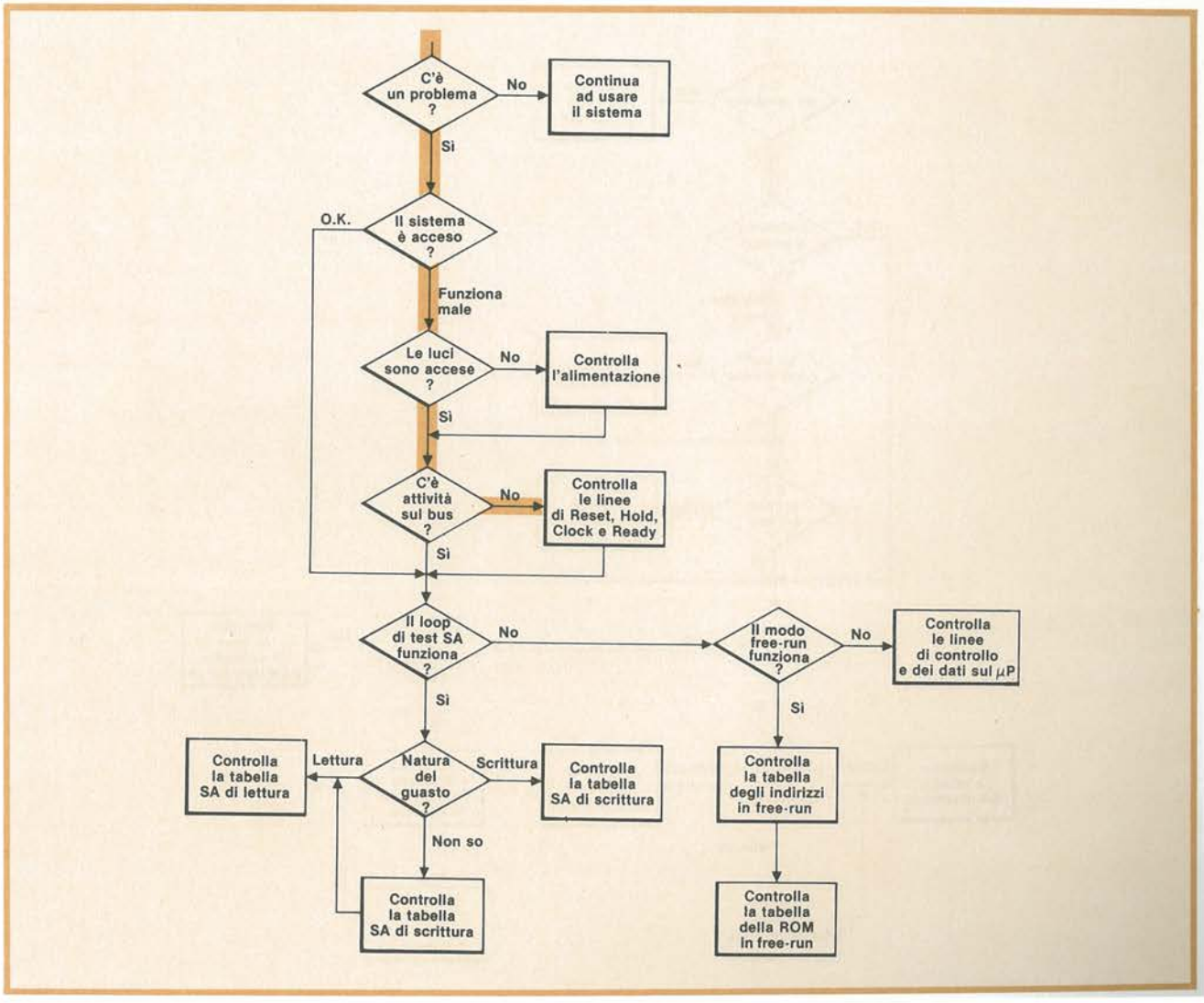
Accensione: Scorretta

Luci accese: Sì

Attività del bus: No

A) La linea di Ready è bloccata bassa in IC3-35.
B) Il segnale di Step (IC10-5), che è collegato alla linea di Ready, è alto.
C) Seguite il percorso del segnale tra IC10-5 e IC3-35 (ingresso Ready del microprocessore).
D) Localizzate il Guasto 7, un corto verso massa sull'ingresso di Ready.

Percorso di ricerca guasti per il Guasto 7



SOLUZIONE DEL GUASTO 8

Problema:

A) Display: Fisso
B) LED d'uscita: Fissi
C) Tastiera: Nessuna risposta

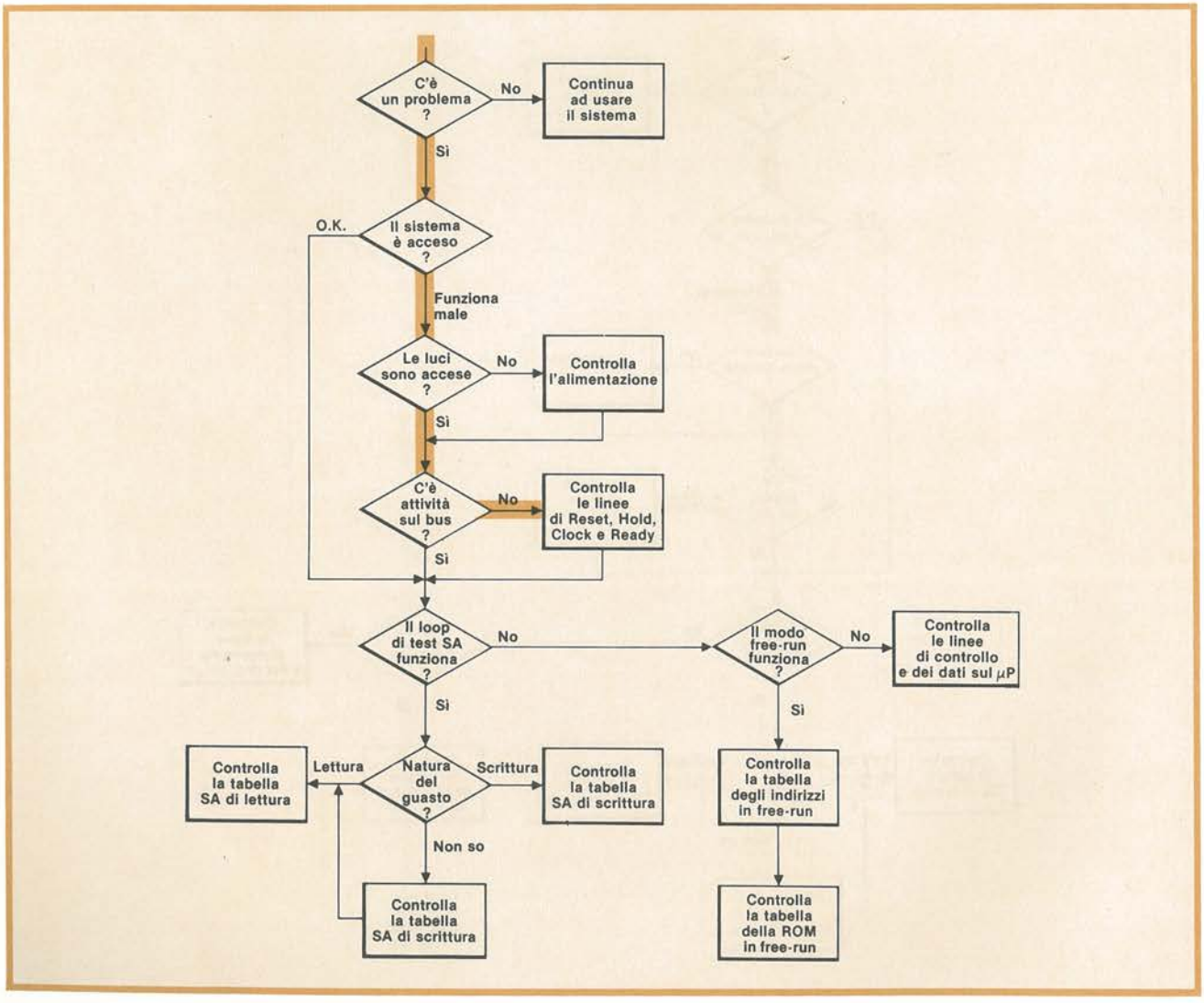
Accensione: Scorretta

Luci accese: Sì

Attività del bus: No

A) La linea di Hold è bloccata bassa su IC3-39.
B) Seguite il segnale da IC3-39 al Guasto 8, un corto verso Vcc.

Percorso di ricerca guasti per il Guasto 8



SOLUZIONE DEL GUASTO 9

Problema:

A) Display: Non si accende il segmento centrale di tutte le cifre del display.
B) LED d'uscita: Il LED D6, all'accensione del sistema, non si illumina.

Accensione: La sequenza di accensione viene portata a termine ma la visualizzazione è scorretta.

Luci accese: Sì

Attività del bus: Sì

Funziona il loop di test SA: Sì

I segmenti centrali del display e il LED di uscita D6 hanno in comune la linea di dato D6 e sono dispositivi verso cui il microprocessore effettua operazioni di scrittura.

Tabella C-3, IC14, 15, 16, 19, display e LED d'uscita: Vengono trovate firme scorrette.

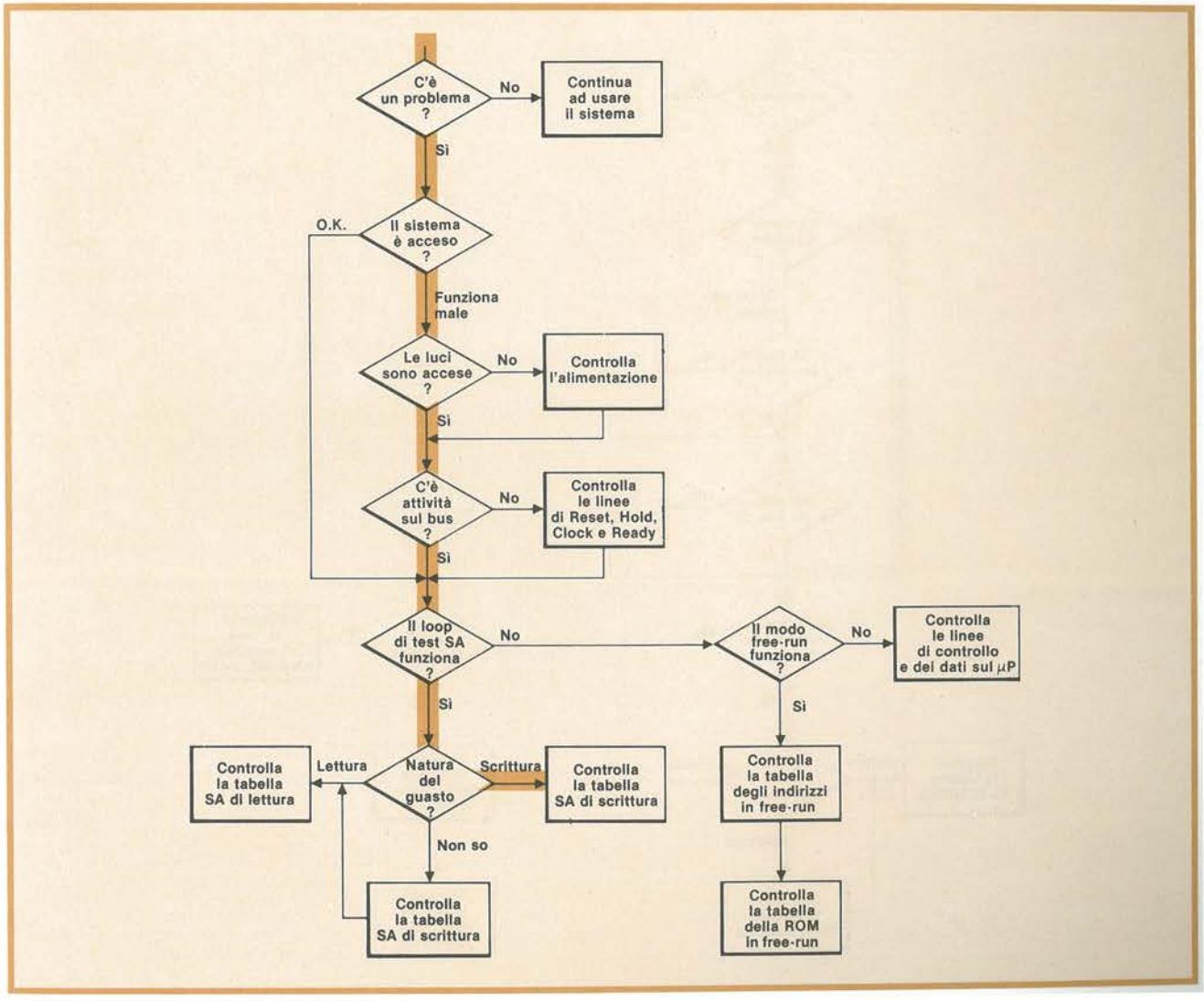
A) La mancanza di attività sul LED d'uscita D6 può essere seguita, risalendo lungo IC15-16 e IC15-17, fino ad ottenere su IC14-15 una firma corretta.

B) La mancanza di attività sul segmento "g" (piedini 7 e 17 del display) può essere seguita, risalendo lungo IC19-11, IC19-7, IC16-16 e IC16-17, fino ad ottenere su IC14-15 una firma corretta.

C) Seguite il percorso del segnale tra IC14-15 e IC16-17.

D) Localizzate il Guasto 9, una pista aperta sulla linea di dato "bufferata" D6.

Percorso di ricerca guasti per il Guasto 9



SOLUZIONE DEL GUASTO 10

Problema:

Quando viene fatto girare il programma dimostrativo ECHO (di indirizzo 04D7) i LED d'uscita 4 e 5 vengono entrambi controllati dall'interruttore d'ingresso 5. L'interruttore d'ingresso 4 non ha alcun effetto.

Accensione: Sì

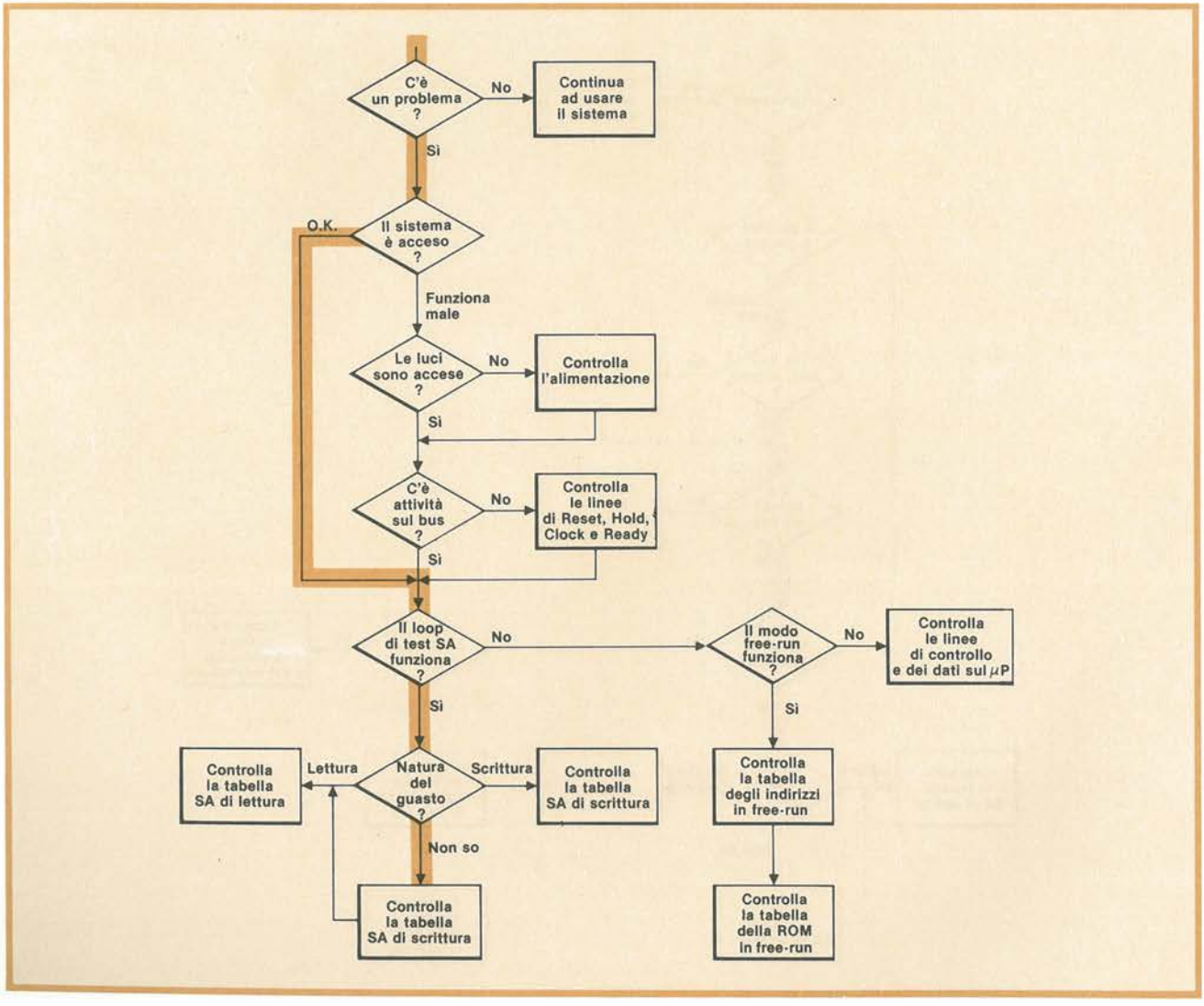
Funziona il loop di test SA: Sì

Potrebbe trattarsi di un guasto nella lettura negli interruttori di ingresso o di un guasto di scrittura sui LED di uscita.

Tabella C-3, IC14-15, e LED d'uscita: Vengono trovate delle firme scorrette.

A) Firme identiche sui LED d'uscita 4 e 5 (la firma dell'uscita 5 è corretta) lasciano presumere che probabilmente esiste un corto tra di esse.
B) Seguite a ritroso le firme, partendo dal LED d'uscita 4 e risalite lungo IC15-12 e IC15-13 fino ad avere una firma corretta su IC14-11.
C) Seguite il percorso del segnale tra IC14-11 e IC15-13.
D) Localizzate il Guasto 10, un corto tra le linee bufferate D4 e D5 del bus dei dati.

Percorso di ricerca guasti per il Guasto 10



SOLUZIONE DEL GUASTO 11

Problema:

Tastiera: Nessuna risposta ai tasti FETCH ADRS, DECR, 3, 6, 9, C e F.

Accensione: Sì

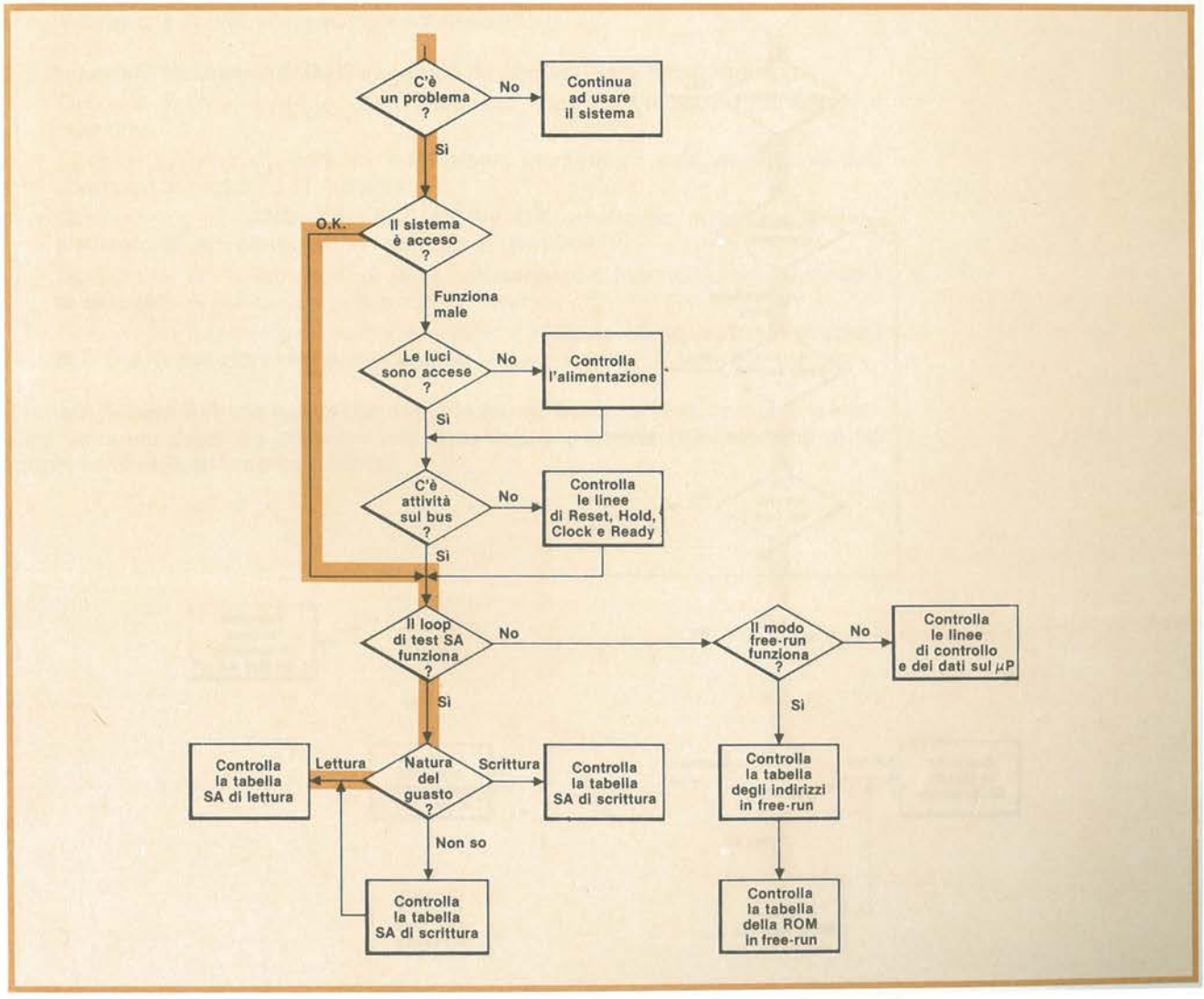
Funziona il loop di test SA: Sì

I tasti che non si riescono a leggere hanno in comune la linea D2. Questo guasto sembra appartenere al gruppo di tasti di lettura.

Tabella C-4, IC18: Vengono trovate delle firme scorrette quando si premono i tasti.

A) Su IC18-15 (D2) le firme non diventano corrette quando viene premuto uno qualunque dei tasti indicati.
B) Quando viene premuto uno di tali tasti, IC18-16 rimane alto.
C) Vicino ai tasti la firma è corretta (seguite le piste sotto i tre tasti, sulla parte alta della scheda).
D) Seguite il percorso del segnale tra la linea della colonna del tasto e IC18-16.
E) Localizzate il Guasto 11, una pista aperta tra la colonna dei tasti 3-F e IC18-16.

Percorso di ricerca guasti per il Guasto 11



SOLUZIONE DEL GUASTO 12

Problema:

A) Display: I due digit di sinistra sono sempre spenti.
B) Tastiera: Premendo i tasti 7, 8 e 9 viene visualizzato 4, 5 e 6 rispettivamente.

Accensione: La sequenza di accensione viene completata ma la visualizzazione è scorretta.
Luci accese: Sì
Attività del bus: Sì

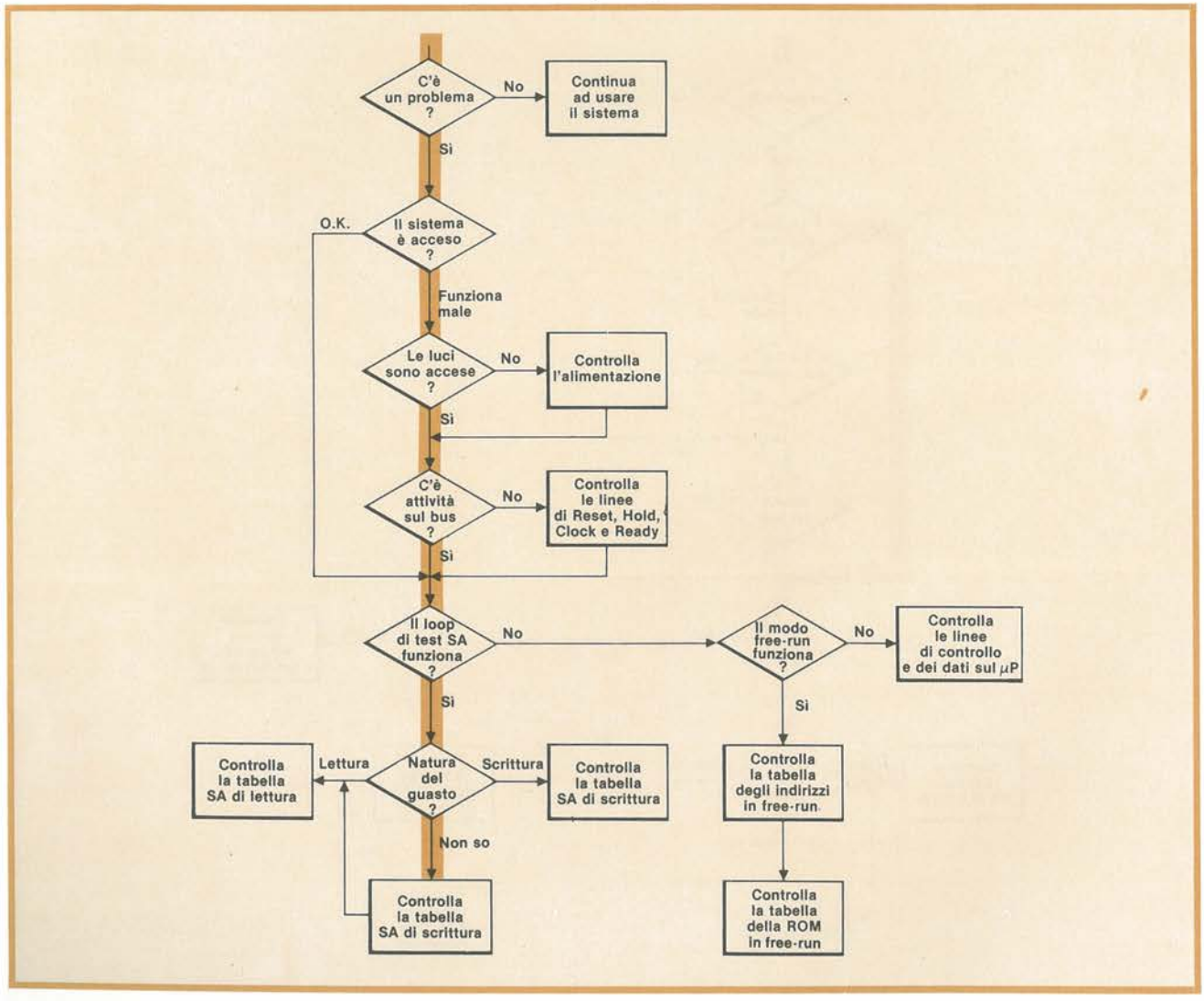
Funziona il loop di test SA: Sì. Stavolta i due digit di sinistra del display si accendono.

Le linee memorizzate di scansione D4 e D5 sono comuni alle parti malfunzionanti del display e della tastiera. Si tratta probabilmente di un guasto di scrittura al latch di scansione IC17.

Tabella C-3, IC17, 20 e tastiera: Vengono trovate delle firme scorrette.

A) Vengono rilevate le stesse firme scorrette su IC20-3 e IC20-5, sulle righe 4-6 e 7-9 di scansione della tastiera e su IC-17-12 e IC17-15.
B) Firme corrette su IC17-13 e IC17-14 indicano che probabilmente è presente un corto tra le linee D4 e D5 in uscita dalla porta di scansione (IC17).
C) Un corto tra i piedini 12 e 15 dovrebbe far passare una corrente abbastanza elevata quando tali piedini d'uscita si trovano in stati logici opposti. Da uno dei piedini d'uscita la corrente dovrebbe fluire alla scheda, sulla pista verso il corto e attraverso il corto, per passare poi sull'altra pista che conduce all'altro piedino d'uscita.
D) Facendo uso del rivelatore di corrente, seguite la corrente sul lato inferiore della scheda da IC17-12 a IC20-3 e ancora al foro metallizzato posto alla sinistra del tasto INTRPT.
E) Notate che la corrente scompare quando il rivelatore di corrente, al di là del foro metallizzato (sempre sul lato inferiore della scheda) viene avvicinato alla tastiera.
F) Seguite la corrente attraverso il foro metallizzato fino al lato superiore della scheda.
G) Localizzate il Guasto 12, un corto tra le linee di scansione 4 e 5.

Percorso di ricerca guasti per il Guasto 12



APPENDICE B

Le istruzioni dell'8085A

INTRODUZIONE

Le informazioni presentate in questa appendice sono state tratte dal manuale dell'Intel "MCS 85™ User's Manual", a cui potete fare riferimento per ulteriori informazioni in merito alle caratteristiche e alle possibilità del Microprocessore 8085. Nel corso della spiegazione delle diverse istruzioni, oltre al formato binario, verrà dato per ogni istruzione anche il codice esadecimale equivalente.

L'insieme delle istruzioni dell'8085 è formato da cinque diversi tipi di istruzioni:

- **Gruppo di trasferimento dati:** muovono i dati tra i registri o tra registri e memorie.
- **Gruppo aritmetico:** somma, sottrazione, incremento e decremento di dati contenuti in registri o in memoria.
- **Gruppo logico:** AND, OR, EXCLUSIVE-OR, confronto, rotazione o complemento di dati contenuti in registri o in memoria.
- **Gruppo di salto:** istruzioni di salto condizionato e incondizionato, istruzioni di chiamata di subroutine e istruzioni di ritorno.
- **Gruppo di controllo macchina, stack e I/O:** ne fanno parte le istruzioni di I/O e le istruzioni che operano sullo stack e sui flag di controllo interno.

Sia nella Tabella B-1 che nelle pagine in cui saranno descritte singolarmente le istruzioni verranno suddivise secondo tale classificazione e nella presentazione di tali gruppi verrà seguito lo stesso ordine.

DATA TRANSFER GROUP

Move

MOV	A,A	7F
	A,B	78
	A,C	79
	A,D	7A
	A,E	7B
	A,H	7C
	A,L	7D
	A,M	7E
MOV	B,A	47
	B,B	40
	B,C	41
	B,D	42
	B,E	43
	B,H	44
	B,L	45
	B,M	46
MOV	C,A	4F
	C,B	48
	C,C	49
	C,D	4A
	C,E	4B
	C,H	4C
	C,L	4D
	C,M	4E
MOV	D,A	57
	D,B	50
	D,C	51
	D,D	52
	D,E	53
	D,H	54
	D,L	55
	D,M	56

Move (cont)

MOV	E,A	5F
	E,B	58
	E,C	59
	E,D	5A
	E,E	5B
	E,H	5C
	E,L	5D
	E,M	5E
MOV	H,A	67
	H,B	60
	H,C	61
	H,D	62
	H,E	63
	H,H	64
	H,L	65
	H,M	66
MOV	L,A	6F
	L,B	68
	L,C	69
	L,D	6A
	L,E	6B
	L,H	6C
	L,L	6D
	L,M	6E
MOV	M,A	77
	M,B	70
	M,C	71
	M,D	72
	M,E	73
	M,H	74
	M,L	75
XCHG		EB

Move Immediate

MVI	A, byte	3E
	B, byte	06
	C, byte	0E
	D, byte	16
	E, byte	1E
	H, byte	26
	L, byte	2E
	M, byte	36

Load Immediate

LXI	B, dble	01
	D, dble	11
	H, dble	21
	SP, dble	31

Load/Store

LDAX B	0A
LDAX D	1A
LHLD adr	2A
LDA adr	3A
STAX B	02
STAX D	12
SHLD adr	22
STA adr	32

byte = constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity (Second byte of 2-byte instructions)
dble = constant, or logical/arithmetic expression that evaluates to a 16-bit data quantity (Second and Third bytes of 3-byte instructions)
adr = 16-bit address (Second and Third bytes of 3-byte instructions)
* = all flags (C, Z, S, P, AC) affected
** = all flags except CARRY affected, (exception: INX and DCX affect no flags)
† = only CARRY affected

ARITHMETIC AND LOGICAL GROUP

Add*

ADD	A	87
	B	80
	C	81
	D	82
	E	83
	H	84
	L	85
	M	86
ADC	A	8F
	B	88
	C	89
	D	8A
	E	8B
	H	8C
	L	8D
	M	8E

Subtract*

SUB	A	97
	B	90
	C	91
	D	92
	E	93
	H	94
	L	95
	M	96
SBB	A	9F
	B	98
	C	99
	D	9A
	E	9B
	H	9C
	L	9D
	M	9E

Double Add †

DAD	B	09
	D	19
	H	29
	SP	39

Increment**

INR	A	3C
	B	04
	C	0C
	D	14
	E	1C
	H	24
	L	2C
	M	34
INX	B	03
	D	13
	H	23
	SP	33

Decrement**

DCR	A	3D
	B	05
	C	0D
	D	15
	E	1D
	H	25
	L	2D
	M	35
DCX	B	0B
	D	1B
	H	2B
	SP	3B

Specials

DAA*	27
CMA	2F
STC†	37
CMC†	3F

Rotate †

RLC	07
RRC	0F
RAL	17
RAR	1F

Logical*

ANA	A	A7
	B	A0
	C	A1
	D	A2
	E	A3
	H	A4
	L	A5
	M	A6
XRA	A	AF
	B	A8
	C	A9
	D	AA
	E	AB
	H	AC
	L	AD
	M	AE
ORA	A	B7
	B	B0
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5
	M	B6
CMP	A	BF
	B	B8
	C	B9
	D	BA
	E	BB
	H	BC
	L	BD
	M	BE

Arith & Logical Immediate

ADI byte	C6
ACI byte	CE
SUI byte	D6
SBI byte	DE
ANI byte	E6
XRI byte	EE
ORI byte	F6
CPI byte	FE

BRANCH CONTROL GROUP

Jump

JMP adr	C3
JNZ adr	C2
JZ adr	CA
JNC adr	D2
JC adr	DA
JPO adr	E2
JPE adr	EA
JP adr	F2
JM adr	FA
PCHL	E9

Call

CALL adr	CD
CNZ adr	C4
CZ adr	CC
CNC adr	D4
CC adr	DC
CPO adr	E4
CPE adr	EC
CP adr	F4
CM adr	FC

Return

RET	C9
RNZ	C0
RZ	C8
RNC	D0
RC	D8
RPO	E0
RPE	E8
RP	F0
RM	F8

Restart

RST	0	C7
	1	CF
	2	D7
	3	DF
	4	E7
	5	EF
	6	F7
	7	FF

I/O AND MACHINE CONTROL

Stack Ops

PUSH	B	C5
	D	D5
	H	E5
	PSW	F5
POP	B	C1
	D	D1
	H	E1
	PSW*	F1
XTHL		E3
SPHL		F9

Input/Output

OUT byte	D3
IN byte	DB

Control

DI	F3
EI	FB
NOP	00
HLT	76

New Instructions (8085 Only)

RIM	20
SIM	30

00	NOP	
01	LXI	B,dble
02	STAX	B
03	INX	B
04	INR	B
05	DCR	B
06	MVI	B,byte
07	RLC	
08	---	
09	DAD	B
0A	LDAX	B
0B	DCX	B
0C	INR	C
0D	DCR	C
0E	MVI	C,byte
0F	RRC	
10	---	
11	LXI	D,dble
12	STAX	D
13	INX	D
14	INR	D
15	DCR	D
16	MVI	D,byte
17	RAL	
18	---	
19	DAD	D
1A	LDAX	D
1B	DCX	D
1C	INR	E
1D	DCR	E
1E	MVI	E,byte
1F	RAR	
20	RIM*	
21	LXI	H,dble
22	SHLD	adr
23	INX	H
24	INR	H
25	DCR	H
26	MVI	H,byte
27	DAA	
28	---	
29	DAD	H
2A	LHLD	adr
2B	DCX	H
2C	INR	L
2D	DCR	L
2E	MVI	L,byte
2F	CMA	
30	SIM*	
31	LXI	SP,dble
32	STA	adr
33	INX	SP
34	INR	M
35	DCR	M
36	MVI	M,byte
37	STC	
38	---	
39	DAD	SP
3A	LDA	adr
3B	DCX	SP
3C	INR	A
3D	DCR	A
3E	MVI	A,byte
3F	CMC	
40	MOV	B,B
41	MOV	B,C
42	MOV	B,D
43	MOV	B,E
44	MOV	B,H
45	MOV	B,L
46	MOV	B,M
47	MOV	B,A
48	MOV	C,B
49	MOV	C,C
4A	MOV	C,D
4B	MOV	C,E
4C	MOV	C,H
4D	MOV	C,L
4E	MOV	C,M
4F	MOV	C,A
50	MOV	D,B
51	MOV	D,C
52	MOV	D,D
53	MOV	D,E
54	MOV	D,H
55	MOV	D,L
56	MOV	D,M
57	MOV	D,A
58	MOV	E,B
59	MOV	E,C
5A	MOV	E,D
5B	MOV	E,E
5C	MOV	E,H
5D	MOV	E,L
5E	MOV	E,M
5F	MOV	E,A
60	MOV	H,B
61	MOV	H,C
62	MOV	H,D
63	MOV	H,E
64	MOV	H,H
65	MOV	H,L
66	MOV	H,M
67	MOV	H,A
68	MOV	L,B
69	MOV	L,C
6A	MOV	L,D
6B	MOV	L,E
6C	MOV	L,H
6D	MOV	L,L
6E	MOV	L,M
6F	MOV	L,A
70	MOV	M,B
71	MOV	M,C
72	MOV	M,D
73	MOV	M,E
74	MOV	M,H
75	MOV	M,L
76	HLT	
77	MOV	M,A
78	MOV	A,B
79	MOV	A,C
7A	MOV	A,D
7B	MOV	A,E
7C	MOV	A,H
7D	MOV	A,L
7E	MOV	A,M
7F	MOV	A,A
80	ADD	B
81	ADD	C
82	ADD	D
83	ADD	E
84	ADD	H
85	ADD	L
86	ADD	M
87	ADD	A
88	ADC	B
89	ADC	C
8A	ADC	D
8B	ADC	E
8C	ADC	H
8D	ADC	L
8E	ADC	M
8F	ADC	A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
96	SUB	M
97	SUB	A
98	SBB	B
99	SBB	C
9A	SBB	D
9B	SBB	E
9C	SBB	H
9D	SBB	L
9E	SBB	M
9F	SBB	A
A0	ANA	B
A1	ANA	C
A2	ANA	D
A3	ANA	E
A4	ANA	H
A5	ANA	L
A6	ANA	M
A7	ANA	A
A8	XRA	B
A9	XRA	C
AA	XRA	D
AB	XRA	E
AC	XRA	H
AD	XRA	L
AE	XRA	M
AF	XRA	A
B0	ORA	B
B1	ORA	C
B2	ORA	D
B3	ORA	E
B4	ORA	H
B5	ORA	L
B6	ORA	M
B7	ORA	A
B8	CMP	B
B9	CMP	C
BA	CMP	D
BB	CMP	E
BC	CMP	H
BD	CMP	L
BE	CMP	M
BF	CMP	A
C0	RNZ	
C1	POP	B
C2	JNZ	adr
C3	JMP	adr
C4	CNZ	adr
C5	PUSH	B
C6	ADI	byte
C7	RST	0
C8	RZ	
C9	RET	
CA	JZ	
CB	---	
CC	CZ	adr
CD	CALL	adr
CE	ACI	byte
CF	RST	1
D0	RNC	
D1	POP	D
D2	JNC	adr
D3	OUT	byte
D4	CNC	adr
D5	PUSH	D
D6	SUI	byte
D7	RST	2
D8	RC	
D9	---	
DA	JC	adr
DB	IN	byte
DC	CC	adr
DD	---	
DE	SBI	byte
DF	RST	3
E0	RPO	
E1	POP	H
E2	JPO	adr
E3	XTHL	
E4	CPO	adr
E5	PUSH	H
E6	ANI	byte
E7	RST	4
E8	RPE	
E9	PCHL	
EA	JPE	adr
EB	XCHG	
EC	CPE	adr
ED	---	
EE	XRI	byte
EF	RST	5
F0	RP	
F1	POP	PSW
F2	JP	adr
F3	DI	
F4	CP	adr
F5	PUSH	PSW
F6	ORI	byte
F7	RST	6
F8	RM	
F9	SPHL	
FA	JM	adr
FB	EI	
FC	CM	adr
FD	---	
FE	CPI	byte
FF	RST	7

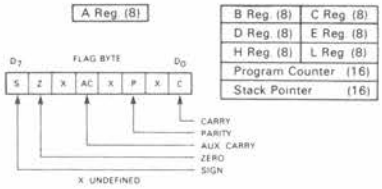
*8085 Only

All mnemonics copyright ® Intel Corporation 1976

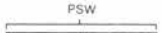
Tabella B-1. Riferimenti per il linguaggio di assemblaggio 8085/8080

INTEL® 8080/8085 INSTRUCTION SET REFERENCE TABLES

INTERNAL REGISTER ORGANIZATION



REGISTER-PAIR ORGANIZATION



NOTE: Leftmost Byte is high-order byte for arithmetic operations and addressing. Left byte is pushed on stack first. Right byte is popped first.

B	(B/C) (16)
D	(D/E) (16)
H	(H/L) (16)
Prog. Ctr.	(16)
Stack Ptr.	(16)

BRANCH CONTROL INSTRUCTIONS

Flag Condition	Jump		Call		Return	
Zero=True	JZ	CA	CZ	CC	RZ	C8
Zero=False	JNZ	C2	CNZ	C4	RNZ	C0
Carry=True	JC	DA	CC	DC	RC	D8
Carry=False	JNC	D2	CNC	D4	RNC	D0
Sign=Positive	JP	F2	CP	F4	RP	F0
Sign=Negative	JM	FA	CM	FC	RM	F8
Parity=Even	JPE	EA	CPE	EC	RPE	E8
Parity=Odd	JPO	E2	CPO	E4	RPO	E0
Unconditional	JMP	C3	CALL	*CD	RET	C9

ACCUMULATOR OPERATIONS

	Code	Function
XRA A	AF	Clear A and Clear Carry
ORA A	B7	Clear Carry
CMC	3F	Complement Carry
CMA	2F	Complement Accumulator
STC	37	Set Carry
RLC	07	Rotate Left
RRC	0F	Rotate Right
RAL	17	Rotate Left Thru Carry
RAR	1F	Rotate Right Thru Carry
DAA	27	Decimal Adjust Accum

RESTART TABLE

Name	Code	Restart Address
RST 0	C7	0000_{16}
RST 1	CF	0008_{16}
RST 2	D7	0010_{16}
RST 3	DF	0018_{16}
RST 4	E7	0020_{16}
TRAP	Hardware* Function	0024_{16}
RST 5	EF	0028_{16}
RST 5.5	Hardware* Function	$002C_{16}$
RST 6	F7	0030_{16}
RST 6.5	Hardware* Function	0034_{16}
RST 7	FF	0038_{16}
RST 7.5	Hardware* Function	$003C_{16}$

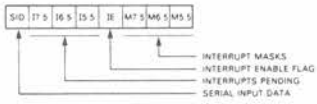
*NOTE: The hardware functions refer to the on-chip Interrupt feature of the 8085 only.

HEX-ASCII TABLE

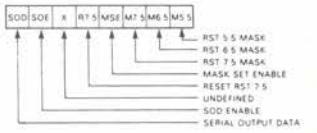
00	NUL		21	!	42	B		63	c
01	SOH		22	"	43	C		64	d
02	STX		23	#	44	D		65	e
03	ETX		24	$	45	E		66	f
04	EOT		25	%	46	F		67	g
05	ENQ		26	&	47	G		68	h
06	ACK		27	'	48	H		69	i
07	BEL		28	(	49	I		6A	j
08	BS		29	)	4A	J		6B	k
09	HT		2A	*	4B	K		6C	l
0A	LF		2B	+	4C	L		6D	m
0B	VT		2C	,	4D	M		6E	n
0C	FF		2D	-	4E	N		6F	o
0D	CR		2E	.	4F	O		70	p
0E	SO		2F	/	50	P		71	q
0F	SI		30	0	51	Q		72	r
10	DLE		31	1	52	R		73	s
11	DC1	(X-ON)	32	2	53	S		74	t
12	DC2	(TAPE)	33	3	54	T		75	u
13	DC3	(X-OFF)	34	4	55	U		76	v
14	DC4	~~(TAPE)~~	35	5	56	V		77	w
15	NAK		36	6	57	W		78	x
16	SYN		37	7	58	X		79	y
17	ETB		38	8	59	Y		7A	z
18	CAN		39	9	5A	Z		7B	{
19	EM		3A	:	5B	[		7C	|
1A	SUB		3B	;	5C	\		7D	}
1B	ESC		3C	<	5D	]			(ALT MODE)
1C	FS		3D	=	5E	^	(↑)	7E	~
1D	GS		3E	>	5F	—	(←)	7F	DEL
1E	RS		3F	?	60	`			(RUB OUT)
1F	US		40	@	61	a			
20	SP		41	A	62	b			

USE OF THE A REGISTER BY RIM AND SIM INSTRUCTIONS (8085 ONLY)

A REGISTER AFTER EXECUTING RIM



A REGISTER BEFORE EXECUTING SIM



REGISTER PAIR AND STACK OPERATIONS

	PSW (A/F)	B (B/C)	D (D/E)	H (H/L)	SP	PC	Function
		Register Pair					
INX		03	13	23	33		Increment Register Pair
DCX		0B	1B	2B	3B		Decrement Register Pair
LDAX		0A	1A	7E(1)			Load A Indirect (Reg. Pair holds Adrs)
STAX		02	12	77(2)			Store A Indirect (Reg. Pair holds Adrs)
LHLD				2A			Load H/L Direct (Bytes 2 and 3 hold Adrs)
SHLD				22			Store H/L Direct (Bytes 2 and 3 hold Adrs)
LXI		01	11	21	31	C3(3)	Load Reg. Pair Immediate (Bytes 2 and 3 hold immediate data)
PCHL						E9	Load PC with H/L (Branch to Adrs in H/L)
XCHG			EB				Exchange Reg. Pairs D/E and H/L
DAD		09	19	29	39		Add Reg. Pair to H/L
PUSH	F5	C5	D5	E5			Push Reg. Pair on Stack
POP	F1	C1	D1	E1			Pop Reg. Pair off Stack
XTHL				E3			Exchange H/L with Top of Stack
SPHL					F9		Load SP with H/L

Notes: 1. This is MOV A,M. 2. This is MOV M,A. 3. This is JMP.

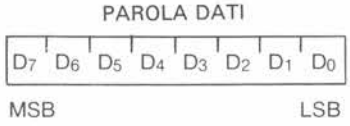
All mnemonics copyright ® Intel Corporation 1976

Tabella B-1. Riferimenti per il linguaggio di assemblaggio 8085/8080 (continuazione)

Formato delle istruzioni e dei dati

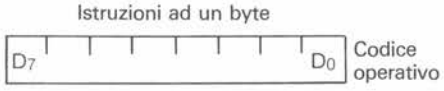
La memoria per l'8085 è oganizzata in quantità di 8 bit, chiamati byte. Ogni byte ha un solo indirizzo binario a 16 bit che corrisponde alla sua posizione sequenziale in memoria. L'8085 può indirizzare direttamente fino a 65.536 byte di memoria, che possono essere costituiti sia da elementi di memoria di sola lettura (ROM) che da elementi di memoria ad accesso casuale (RAM) (memoria di lettura / scrittura).

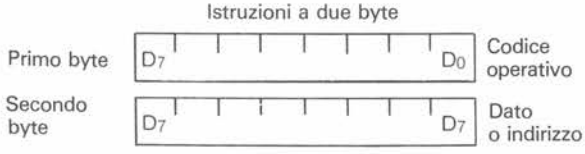
I dati nell'8085 vengono memorizzati sotto forma di numeri interi binari a 8 bit:

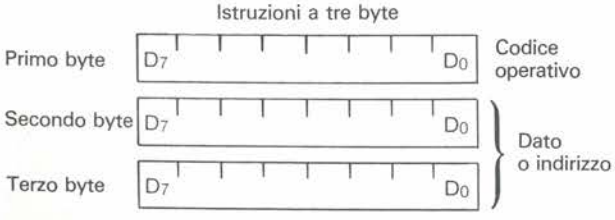


Quando un registro o una parola dati contiene un numero binario, è necessario stabilire l'ordine nel quale sono scritti i bit del numero. Nell'8085, ci si riferisce al BIT 0 come al **Bit meno significativo (LSB)** ed al BIT 7 (di un numero a 8 bit) come al **Bit più significativo (MSB)**.

Le istruzioni di programma dell'8085 possono essere di uno, due o tre byte di lunghezza. Le istruzioni a più byte devono essere memorizzate in successive posizioni di memoria; l'indirizzo del primo byte viene sempre usato come indirizzo delle istruzioni. L'esatto formato dell'istruzione dipenderà dalla particolare operazione che deve essere eseguita.







Modi di indirizzamento

Spesso i dati sui quali bisogna operare sono immagazzinati in memoria. Quando si usano i dati numerici a più byte, i dati, come le istruzioni, vengono memorizzati in posizioni di memoria successive, con il byte meno significativo al primo posto, seguito via via dai byte più significativi. L'8085 ha quattro modi diversi di indirizzare i dati caricati in memoria o nei registri:

- **Diretto**: I byte 2 e 3 dell'istruzione contengono l'esatto indirizzo di memoria dei dati in questione (i bit di indirizzo di ordine basso sono nel byte 2, i bit di ordine più alto sono nel byte 3).
- **Registro**: L'istruzione specifica il registro e la coppia di registri in cui i dati sono posizionati.
- **Registro indiretto**: L'istruzione specifica una coppia di registri che contiene l'indirizzo di memoria in cui i dati sono posizionati (i bit di ordine più alto sono nel primo dei due registri, i bit di ordine basso nel secondo).
- **Immediato**: L'istruzione contiene i dati stessi. Questi sono in quantità di 8 o di 16 bit (il byte meno significativo per primo, il byte più significativo per secondo).

A meno che non venga diretta da un'istruzione d'interruzione o di salto, l'esecuzione delle istruzioni procede attraverso posizioni di memoria che aumentano sequenzialmente. Un'istruzione di salto può specificare l'indirizzo della istruzione successiva che deve essere eseguita in uno dei due modi seguenti:

- **Diretto**: L'istruzione di salto contiene l'indirizzo dell'istruzione seguente che va eseguita. Eccetto che per l'istruzione «RST», il byte 2 contiene l'indirizzo di ordine basso ed il byte 3 l'indirizzo di ordine più alto.
- **Registro indiretto**: L'istruzione di salto indica una coppia di registri che contiene l'indirizzo dell'istruzione che va eseguita successivamente. I bit d'indirizzo di ordine più alto sono nel primo dei due registri, i bit di ordine basso nel secondo.

L'istruzione RST è una speciale istruzione chiamata ad un byte (usata di solito durante le sequenze d'interruzione). RST contiene un campo a 3 bit; il controllo del programma viene trasferito all'istruzione il cui indirizzo è otto volte il contenuto di questo campo a tre bit.

Tutti i mnemonici sono registrati ® Intel Corporation 1976

Flag di condizione (Condition Flag)

Ci sono cinque flag di condizione collegati con l'esecuzione delle istruzioni sull'8085. Essi sono Zero, Sign, Parity, Carry e Auxiliary Carry, ed ognuno di loro è rappresentato da un registro ad 1 bit nella CPU. Un flag viene «settato» forzando il bit a 1, «resettato» forzando il bit a 0.

Salvo diversa indicazione, quando un'istruzione altera un flag, lo fa nel modo seguente:

- **Zero**: se il risultato di un'istruzione ha il valore 0, questo flag è settato, altrimenti viene resettato.

- **Sign** (segno): se il bit più significativo del risultato dell'operazione ha il valore 1, questo flag è settato, altrimenti viene resettato.

- **Parity** (parità): se la somma modulo 2 dei bit del risultato della operazione è 0, (cioè se il risultato ha parità pari), questo flag viene settato, altrimenti viene resettato (cioè se il risultato ha parità dispari).

- **Carry** (riporto): se l'istruzione determina l'uscita di un riporto (da addizione), o di un riporto negativo (da sottrazione o da un confronto) dal bit più significativo, questo flag viene settato altrimenti viene resettato.

- **Auxiliary Carry** (riporto ausiliario): se l'istruzione ha dato luogo ad un riporto dal bit 3 al bit 4 del valore risultante, il riporto ausiliario viene settato, altrimenti è resettato. Questo flag riguarda le addizioni, le sottrazioni, gli incrementi, i decrementi, i confronti e le operazioni logiche, ma viene usato principalmente con le addizioni e gli incrementi che precedono un'istruzione DAA (Decimal Adjust Accumulator, Aggiustamento decimale dell'accumulatore).

Tutti i mnemonici sono registrati ® Intel Corporation 1976

Simboli ed abbreviazioni

I simboli e le abbreviazioni che seguono vengono usati nella descrizione successiva delle istruzioni dell'8085:

SIMBOLI	SIGNIFICATO
accumulator	Registro A
addr	Quantità di indirizzo a 16 bit
byte	Quantità di dati ad 8 bit
dble	Quantità di dati a 16 bit
byte 2	Il secondo byte dell'istruzione
byte 3	Il terzo byte dell'istruzione
port	Indirizzo ad 8 bit di un dispositivo di I/O
r, r1, r2	Uno dei registri A,B,C,D,E,H,L
DDD, SSS	La configurazione di bit che designa, uno dei registri A,B,C,D,E, H,L (DDD = destinazione, SSS = sorgente):

DDD o SSS	NOME DEL REGISTRO
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp — Una delle coppie di registri:

B rappresenta la coppia B,C dove B è il registro di ordine alto e C il registro di ordine basso;

D rappresenta la coppia D, E dove D è il registro di ordine alto ed E il registro di ordine basso;

H rappresenta la coppia H,L dove H è il registro di ordine alto ed L il registro di ordine basso;

SP rappresenta il registro puntatore dello stack a 16 bit.

SIMBOLI	SIGNIFICATO
RP	La configurazione di bit che designa una delle coppie di registri B,D,H,SP:

RP	COPPIA DI REGISTRI
00	B-C
01	D-E
10	H-L
11	SP

rh	Il primo registro (ordine alto) di una certa coppia di registri.
rl	Il secondo registro (ordine basso) di una certa coppia di registri.
PC	Registro contatore di programma a 16 bit (PCH e PCL vengono usati per riferirsi rispettivamente agli 8 bit di ordine più alto ed a quelli di ordine basso).
SP	Registro puntatore dello stack a 16 bit (SPH e SPL vengono usati per riferirsi rispettivamente agli 8 bit di ordine alto ed a quelli di ordine basso).
r_m	Bit m del registro r (i bit vanno dal numero 7 allo 0, da sinistra verso destra).
Z,S,P,CY,AC	I flag di condizione: Zero, Sign, Parity, Carry ed Auxiliary Carry, rispettivamente.
( )	Il contenuto della posizione di memoria o dei registri indicati entro la parentesi
←	«Viene trasferito a»
∧	AND logico
⊻	OR esclusivo
∨	OR inclusivo
+	Addizione
—	Sottrazione in complemento di due
*	Moltiplicazione
↔	«Viene scambiato con»
$\overline{}$	Il complemento ad uno (per esempio $(\overline{A})$)
n	Numeri da 0 a 7
NNN	La rappresentazione binaria da 000 a 111 dei numeri di restart da 0 a 7 rispettivamente

Descrizione del formato

Le pagine seguenti presentano una descrizione dettagliata dell'insieme di istruzioni dell'8085. Ogni istruzione è descritta nel modo seguente:

1. Il formato dell'istruzione è costituito dal codice mnemonico di istruzione e dai campi operandi, ed è stampato in **NERETTO** sul lato sinistro della prima riga. Notate che prima di poter essere introdotto nel Microprocessor Lab ogni codice deve essere convertito in esadecimale.

2. Il nome dell'istruzione è chiuso fra parentesi sul lato destro della prima riga.

3. La riga o le righe successive contengono una descrizione simbolica dell'operazione dell'istruzione.

4. Segue quindi una descrizione per esteso dell'operazione dell'istruzione.

5. La riga o le righe seguenti contengono le configurazioni ed i campi binari che comprendono l'istruzione macchina.

6. Le linee seguenti contengono il codice operativo in formato esadecimale evidenziato in colore.

7. Le ultime quattro righe contengono informazioni varie circa l'esecuzione dell'istruzione. Per primo è elencato il numero di stati e di cicli macchina richiesto per eseguire l'istruzione. Se l'istruzione ha due tempi di esecuzione possibili, come nel caso dei salti condizionali, saranno indicati tutti e due i tempi, separati da una barra. Quindi, sono mostrati tutti i modi significativi di indirizzamento dei dati. L'ultima riga indica alcuni dei cinque flag che sono coinvolti nell'esecuzione dell'istruzione.

Tutti i mnemonici sono registrati ® Intel Corporation 1976

Gruppo trasferimento dati

Questo gruppo di istruzioni trasferisce i dati da e verso i registri e la memoria. In questo gruppo i flag di condizione non vengono alterati da nessuna istruzione.

MOV r1, r2 (Spostare il registro)
(r1) ← (r2)
Il contenuto del registro r2 è spostato nel registro r1

0	1	D	D	D	S	S	S

Formato esadecimale

MOV	A, A	7F
MOV	A, B	78
MOV	A, C	79
MOV	A, D	7A
MOV	A, E	7B
MOV	A, H	7C
MOV	A, L	7D
MOV	B, A	47
MOV	B, B	40
MOV	B, C	41
MOV	B, D	42
MOV	B, E	43
MOV	B, H	44
MOV	B, L	45
MOV	C, A	4F
MOV	C, B	48
MOV	C, C	49
MOV	C, D	4A
MOV	C, E	4B
MOV	C, H	4C
MOV	C, L	4D
MOV	D, A	57
MOV	D, B	50
MOV	D, C	51
MOV	D, D	52
MOV	D, E	53
MOV	D, H	54
MOV	D, L	55
MOV	E, A	5F
MOV	E, B	58
MOV	E, C	59
MOV	E, D	5A
MOV	E, E	5B
MOV	E, H	5C
MOV	E, L	5D
MOV	H, A	67
MOV	H, B	60
MOV	H, C	61
MOV	H, D	62
MOV	H, E	63
MOV	H, H	64
MOV	H, L	65
MOV	L, A	6F
MOV	L, B	68
MOV	L, C	69
MOV	L, D	6A
MOV	L, E	6B
MOV	L, H	6C
MOV	L, L	6D

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: nessuno

MOV r,M (Spostare dalla memoria)
(r) ← ((H) (L))
Il contenuto della posizione di memoria, il cui indirizzo è nei registri H ed L, viene spostato nel registro r.

0	1	D	D	D	1	1	0

Formato esadecimale

MOV	A, M	7E
MOV	B, M	46
MOV	C, M	4E
MOV	D, M	56
MOV	E, M	5E
MOV	H, M	66
MOV	L, M	6E

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

MOV M,r (Spostare verso la memoria)

((H) (L)) ← (r)

Il contenuto del registro r viene spostato verso la posizione di memoria il cui indirizzo è nei registri H e L.

0	1	1	1	0	S	S	S

Formato esadecimale

MOV	M, A	77
MOV	M, B	70
MOV	M, C	71
MOV	M, D	72
MOV	M, E	73
MOV	M, H	74
MOV	M, L	75

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: nessuno

MVI r, byte (Spostare in modo immediato)

(r) ← (byte 2)

Il contenuto del byte 2 dell'istruzione viene spostato nel registro r.

0	0	D	D	D	1	1	0
dato							

Formato esadecimale

MVI	A, byte	3E
MVI	B, byte	06
MVI	C, byte	0E
MVI	D, byte	16
MVI	E, byte	1E
MVI	H, byte	26
MVI	L, byte	2E

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: nessuno

MVI M, byte (Spostare in modo immediato verso la memoria)

((H) (L)) ← (byte 2)

Il contenuto del byte 2 dell'istruzione viene spostato nella posizione di memoria il cui indirizzo è nei registri H ed L.

0	0	1	1	0	1	1	0
dato							

Formato esadecimale

MVI	M, byte	36

Cicli: 3
Stati: 10
Indirizzamento: immediato / reg. indiretto
Flag: nessuno

LXI rp, dble (Caricare in modo immediato la coppia di registri)

(rh) ← (byte 3)

(rl) ← (byte 2)

Il byte 3 dell'istruzione viene spostato nel registro di ordine alto (rh) della coppia di registri rp. Il byte 2 dell'istruzione viene spostato nel registro di ordine basso (rl) della coppia di registri rp.

0	0	R	P	0	0	0	1
dato (ordine basso)							
dato (ordine alto)							

Formato esadecimale

LXI	B, dble	01	(caricare in modo immediato la coppia di registri B e C).
LXI	D, dble	11	(caricare in modo immediato la coppia di registri D e E).
LXI	H, dble	21	(caricare in modo immediato la coppia di registri H e L).
LXI	SP, dble	31	(caricare in modo immediato il puntatore dello stack).

Cicli: 3
Stati: 10
Indirizzamento: immediato
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

LDA addr (Caricare direttamente l'accumulatore)

(A) ← ((byte 3) (byte 2))

Il contenuto della posizione di memoria, il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione, viene spostato nel registro A.

0	0	1	1	1	0	1	0
indirizzo (ordine basso)							
indirizzo (ordine alto)							

Formato esadecimale

LDA addr 3A

Cicli: 4
Stati: 13
Indirizzamento: diretto
Flag: nessuno

LHLD addr (Caricare direttamente H ed L)

(L) ← ((byte 3) (byte 2))

(H) ← ((byte 3) (byte 2) + 1)

Il contenuto della posizione di memoria, il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione, viene spostato nel registro L. Il contenuto della posizione di memoria all'indirizzo successivo viene spostato nel registro H.

0	0	1	0	1	0	1	0
indirizzo (ordine basso)							
indirizzo (ordine alto)							

Formato esadecimale

LHLD addr 2A

Cicli: 5
Stati: 16
Indirizzamento: diretto
Flag: nessuno

STA addr (Memorizzare direttamente l'accumulatore)

((byte 3) (byte 2)) ← (A)

Il contenuto dell'accumulatore viene spostato nella posizione di memoria il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione.

0	0	1	1	0	0	1	0
indirizzo (ordine basso)							
indirizzo (ordine alto)							

Formato esadecimale

STA addr 32

Cicli: 4
Stati: 13
Indirizzamento: diretto
Flag: nessuno

SHLD addr (Memorizzare direttamente H ed L)

((byte 3) (byte 2)) ← (L)

((byte 3) (byte 2) + 1) ← (H)

Il contenuto del registro L viene spostato nella posizione di memoria il cui indirizzo è specificato nei byte 2 e 3. Il contenuto del registro H viene spostato nella posizione di memoria successiva.

0	0	1	0	0	0	1	0
indirizzo (ordine basso)							
indirizzo (ordine alto)							

Formato esadecimale

SHLD addr 22

Cicli: 5
Stati: 16
Indirizzamento: diretto
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

LDAX rp (Caricare indirettamente l'accumulatore)

(A) ← ((rp))

Il contenuto della posizione di memoria, il cui indirizzo è nella coppia di registri rp, viene spostato nel registro A. Nota: solo le coppie di registri rp = B (registri B e C) o rp = D (registri D e E) possono essere specificate.

0	0	R	P	1	0	1	0

Formato esadecimale

LDAX B 0A
LDAX D 1A

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: nessuno

STAX rp (Memorizzare indirettamente l'accumulatore)

((rp)) ← (A)

Il contenuto del registro A viene spostato nella posizione di memoria il cui indirizzo è nella coppia di registri rp. Nota: solo le coppie di registri rp = B (registri B e C) o rp = D (registri D e E) possono essere specificate.

0	0	R	P	0	0	1	0

Formato esadecimale

STAX B 02
STAX D 12

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: nessuno

XCHG (Scambiare H ed L con D ed E)

(H) ↔ (D)
(L) ↔ (E)

Il contenuto dei registri H ed L viene scambiato con i contenuti dei registri D ed E.

1	1	1	0	1	0	1	1

Formato esadecimale

XCHG EB

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

Gruppo aritmetico

Questo gruppo di istruzioni esegue operazioni aritmetiche su dati nei registri e nella memoria.

Salvo diversa indicazione, tutte le istruzioni di questo gruppo coinvolgono i flag Zero, Parity, Carry e Auxiliary Carry secondo le regole standard.

Tutte le operazioni di sottrazione sono eseguite per mezzo dell'aritmetica complemento a due e portano il carry ad uno per indicare un riporto negativo, e lo azzerano se non vi è nessun riporto negativo.

ADD r (Addizionare il registro)
(A) ← (A) + (r)
Il contenuto del registro r è sommato al contenuto dell'accumulatore. Il risultato è posto nell'accumulatore.

1	0	0	0	0	S	S	S

Formato esadecimale

ADD A 87
ADD B 80
ADD C 81
ADD D 82
ADD E 83
ADD H 84
ADD L 85

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

ADD M (Addizionare la memoria)
(A) ← (A) + ((H) (L))
Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L è sommato al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

1	0	0	0	0	1	1	0

Formato esadecimale

ADD M 86

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

ADI byte (Addizionare in modo immediato)
(A) ← (A) + (byte 2)
Il contenuto del secondo byte dell'istruzione è sommato al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

1	1	0	0	0	1	1	0
dato							

Formato esadecimale

ADI byte C6

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

ADC r (Addizionare il registro e il riporto)

(A) ← (A) + (r) + (CY)

Il contenuto del registro r ed il contenuto del bit di carry sono sommati al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

1	0	0	0	1	S	S	S

Formato esadecimale

ADC	A	8F
ADC	B	88
ADC	C	89
ADC	D	8A
ADC	E	8B
ADC	H	8C
ADC	L	8D

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

ACI byte (Addizionare in modo immediato con riporto)

(A) ← (A) + (byte 2) + (CY)

Il contenuto del secondo byte dell'istruzione ed il contenuto del flag CY sono sommati al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

1	1	0	0	1	1	1	0
dato							

Formato esadecimale

ACI byte CE

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

ADC M (Addizionare la memoria e il riporto)

(A) ← (A) + ((H) (L)) + (CY)

Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L ed il contenuto del flag CY sono sommati all'accumulatore. Il risultato si trova nell'accumulatore.

1	0	0	0	1	1	1	0

Formato esadecimale

ADC M 8E

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

SUB r (Sottrarre il registro)

(A) ← (A) — (r)

Il contenuto del registro r è sottratto dal contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

1	0	0	1	0	S	S	S

Formato esadecimale

SUB	A	97
SUB	B	90
SUB	C	91
SUB	D	92
SUB	E	93
SUB	H	94
SUB	L	95

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

SUB M (Sottrarre la memoria)

(A) ← (A) — ((H) (L))

Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L è sottratto al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

1	0	0	1	0	1	1	0

Formato esadecimale

SUB M 96

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

SBB r (Sottrarre il registro con riporto negativo)

(A) ← (A) — (r) — (CY)

Il contenuto del registro r ed il contenuto del flag CY vengono sottratti entrambi all'accumulatore. Il risultato si trova nell'accumulatore.

1	0	0	1	1	S	S	S

Formato esadecimale

SBB A 9F
SBB B 98
SBB C 99
SBB D 9A
SBB E 9B
SBB H 9C
SBB L 9D

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

SUI byte (Sottrarre in modo immediato)

(A) ← (A) — (byte 2)

Il contenuto del secondo byte dell'istruzione è sottratto al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore

1	1	0	1	0	1	1	0
dato							

Formato esadecimale

SUI byte D6

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

SBB M (Sottrarre la memoria con riporto negativo)

(A) ← (A) — ((H) (L)) — (CY)

Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H e L ed il contenuto del flag CY sono entrambi sottratti all'accumulatore. Il risultato si trova nell'accumulatore.

1	0	0	1	1	1	1	0

Formato esadecimale

SBB M 9E

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

SBI byte (Sottrarre in modo immediato con riporto negativo)

(A) ← (A) — (byte 2) — (CY)

Il contenuto del secondo byte dell'istruzione ed il contenuto del flag CY vengono entrambi sottratti dall'accumulatore. Il risultato si trova nell'accumulatore.

1	1	0	1	1	1	1	0
dato							

Formato esadecimale

SBI byte DE

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

INR M (Incrementare la memoria)

((H) (L)) ← ((H) (L)) + 1

Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L è incrementato di uno. Nota: tutti i flag di condizione, eccetto CY, sono coinvolti.

0	0	1	1	0	1	0	0

Formato esadecimale

INR M 34

Cicli: 3
Stati: 10
Indirizzamento: registro indiretto
Flag: Z,S,P,AC

INR r (Incrementare il registro)

(r) ← (r) + 1

Il contenuto del registro r è incrementato di uno. Nota: tutti i flag di condizione, eccetto CY, sono coinvolti.

0	0	D	D	D	1	0	0

Formato esadecimale

INR A 3C
INR B 04
INR C 0C
INR D 14
INR E 1C
INR H 24
INR L 2C

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,AC

DCR r (Decrementare il registro)

(r) ← (r) — 1

Il contenuto del registro r è decrementato di uno. Nota: tutti i flag di condizione, eccetto CY, sono coinvolti.

0	0	D	D	D	1	0	1

Formato esadecimale

DCR A 3D
DCR B 05
DCR C 0D
DCR D 15
DCR E 1D
DCR H 25
DCR L 2D

Cicli: 1
Stati: 4
Indirizzamento: registro indiretto
Flag: Z,S,P,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

DCR M (Decrementare la memoria)
((H) (L)) ← ((H) (L)) — 1
Il contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H e L è decrementato di uno. Nota: tutti i flag di condizione, eccetto CY, sono coinvolti.

0	0	1	1	0	1	0	1

Formato esadecimale

DCR M 35

Cicli: 3
Stati: 10
Indirizzamento: registro indiretto
Flag: Z,S,P,AC

DCX rp (Decrementare la coppia di registri)
(rh) (rl) ← (rh) (rl) — 1
Il contenuto della coppia di registri rp è decrementato di uno. Nota: nessun flag di condizione è coinvolto.

0	0	R	P	1	0	1	1

Formato esadecimale

DCX	B	0B	(coppia di registri B/C)
DCX	D	1B	(coppia di registri D/E)
DCX	H	2B	(coppia di registri H/L)
DCX	SP	3B	(puntatore di stack)

Cicli: 1
Stati: 6
Indirizzamento: registro
Flag: nessuno

INX rp (Incrementare la coppia di registri)
(rh) (rl) ← (rh) (rl) + 1
Il contenuto della coppia di registri rp è incrementato di uno. Nota: nessun flag di condizione è coinvolto.

0	0	R	P	0	0	1	1

Formato esadecimale

INX	B	03	(coppia di registri B/C)
INX	D	13	(coppia di registri D/E)
INX	H	23	(coppia di registri H/L)
INX	SP	33	(puntatore di stack)

Cicli: 1
Stati: 6
Indirizzamento: registro
Flag: nessuno

DAD rp (Addizionare la coppia di registri a H e L)
(H) (L) ← (H) (L) + (rh) (rl)
Il contenuto della coppia di registri rp è sommato al contenuto della coppia di registri H ed L. Il risultato si trova nella coppia di registri H ed L. Nota: solo il flag CY è coinvolto. Esso è settato se vi è un riporto dall'addizione in doppia precisione, altrimenti è resettato.

0	0	R	P	1	0	0	1

Formato esadecimale

DAD	B	09	(coppia di registri B/C)
DAD	D	19	(coppia di registri D/E)
DAD	H	29	(coppia di registri H/L)
DAD	SP	39	(puntatore di stack)

Cicli: 3
Stati: 10
Indirizzamento: registro
Flag: CY

Tutti i mnemonici sono registrati ® Intel Corporation 1976

DAA (Aggiustamento decimale dell'accumulatore)

Il numero ad otto bit nell'accumulatore viene adattato per formare due cifre decimali in codice BCD per mezzo del seguente procedimento:

1. Se il valore dei 4 bit meno significativi dell'accumulatore è maggiore di 9 o se il flag AC è settato, si somma 6 all'accumulatore.
2. Se il valore dei 4 bit più significativi dell'accumulatore è ora maggiore di 9, o se il flag CY è settato, si somma 6 ai 4 bit più significativi dell'accumulatore.

Nota: tutti i flag sono coinvolti

0	0	1	0	0	1	1	1

Formato esadecimale

DAA 27

Cicli: 1
Stati: 4
Flag: Z,S,P,CY,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

Gruppo logico

Questo gruppo di istruzioni esegue operazioni logiche, (boolean) sui dati nei registri e nella memoria e sui flag di condizione.

Salvo diversa indicazione, tutte le istruzioni di questo gruppo coinvolgono i flag Zero, Sign, Parity, Carry, Auxiliary Carry secondo le regole solite.

ANA r (AND del registro)

(A) ← (A) ∧ (r)

Il contenuto del registro r è posto in AND logico con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **Il flag CY è azzerato mentre AC è posto a 1**

1	0	1	0	0	S	S	S

Formato esadecimale

ANA A A7
ANA B A0
ANA C A1
ANA D A2
ANA E A3
ANA H A4
ANA L A5

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

ANA M (AND della memoria)

(A) ← (A) ∧ ((H) (L))

Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L è posto in AND logico con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **Il flag CY è azzerato, mentre AC è posto a 1.**

1	0	1	0	0	1	1	0

Formato esadecimale

ANA M A6

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

ANI byte (AND in modo immediato)

(A) ← (A) ∧ (byte 2)

Il contenuto del secondo byte dell'istruzione è posto in AND logico con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **Il flag CY è azzerato, mentre il flag AC è posto a 1.**

1	1	1	0	0	1	1	0
dato							

Formato esadecimale

ANI byte E6

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

XRA r (OR esclusivo del registro)

(A) ← (A) ∀ (r)

Viene eseguita un'operazione di OR esclusivo tra il contenuto del registro r e il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

1	0	1	0	1	S	S	S

XRA	A	AF
XRA	B	A8
XRA	C	A9
XRA	D	AA
XRA	E	AB
XRA	H	AC
XRA	L	AD

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

XRI byte (OR esclusivo in modo immediato)

(A) ← (A) ∀ (byte 2)

Viene eseguita un'operazione di OR esclusivo tra il contenuto del secondo byte della istruzione e il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

1	1	1	0	1	1	1	0
dato							

Formato esadecimale

XRI byte EE

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

XRA M (OR esclusivo della memoria)

(A) ← (A) ∀ ((H) (L))

Viene eseguita un'operazione di OR esclusivo tra il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L e il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

1	0	1	0	1	1	1	0

Formato esadecimale

XRA M AE

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

ORA r (OR del registro)

(A) ← (A) ∨ (r)

Viene eseguita un'operazione di OR inclusivo tra il contenuto del registro r e il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

I flag CY e AC vengono azzerati.

1	0	1	1	0	S	S	S

Formato esadecimale

ORA	A	B7
ORA	B	B0
ORA	C	B1
ORA	D	B2
ORA	E	B3
ORA	H	B4
ORA	L	B5

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

ORA M (OR della memoria)
(A) ← (A) ∨ ((H) (L))
Viene eseguita un'operazione di OR inclusivo tra il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L ed il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

1	0	1	1	0	1	1	0

Formato esadecimale

ORA M B6

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

ORI byte (OR in modo immediato)
(A) — (A) ∨ (byte 2)
Viene eseguita un'operazione di OR inclusivo tra il contenuto del secondo byte dell'istruzione e il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

1	1	1	1	0	1	1	0
dato							

Formato esadecimale

ORI byte F6

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

CMP r (Confrontare il registro)
(A) — (r)
Il contenuto del registro r è sottratto all'accumulatore. L'accumulatore resta invariato. I flag di condizione vengono modificati a seconda del risultato della sottrazione. **Il flag Z è posto a 1 se (A) = (r). Il flag CY è posto a 1 se (A) < (r).**

1	0	1	1	1	S	S	S

Formato esadecimale

CMP A BF
CMP B B8
CMP C B9
CMP D BA
CMP E BB
CMP H BC
CMP L BD

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

CMP M (Confrontare la memoria)
(A) — ((H) (L))
Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L viene sottratto all'accumulatore. L'accumulatore resta invariato. I flag sono modificati a seconda del risultato della sottrazione. **Il flag Z è posto a 1 se (A) = ((H) (L)). Il flag CY è posto a 1 se (A) < ((H) (L)).**

1	0	1	1	1	1	1	0

Formato esadecimale

CMP M BE

Cicli: 2
Stati: 7
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

CPI byte (Confrontare in modo immediato)

(A) — (byte 2)

Il contenuto del secondo byte dell'istruzione è sottratto all'accumulatore. I flag sono modificati a seconda del risultato della sottrazione. **Il flag Z è posto a 1 se (A) = (byte 2). Il flag CY è posto a 1 se (A) < (byte 2).**

1	1	1	1	1	1	1	0
dato							

Formato esadecimale

CPI byte FE

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

RRC (Ruotare a destra)

$(A_n) \leftarrow (A_{n+1})$; $(A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

Il contenuto dell'accumulatore viene fatto ruotare a destra di una posizione. Il bit di ordine più alto ed il flag CY sono caricati entrambi con il valore spostatosi dal bit di ordine basso. **È coinvolto solo il flag CY.**

0	0	0	0	1	1	1	1

Formato esadecimale

RRC 0F

Cicli: 1
Stati: 4
Flag: CY

RLC (Ruotare a sinistra)

$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$;

$(CY) \leftarrow (A_7)$

Il contenuto dell'accumulatore viene fatto ruotare a sinistra di una posizione. Il bit di ordine basso e il flag CY sono caricati entrambi con il valore spostatosi dal bit di ordine più alto. **È coinvolto solo il flag CY.**

0	0	0	0	0	1	1	1

Formato esadecimale

RLC 07

Cicli: 1
Stati: 4
Flag: CY

RAL (Ruotare a sinistra attraverso il riporto)

$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$

$(A_0) \leftarrow (CY)$

Il contenuto dell'accumulatore viene fatto ruotare a sinistra di una posizione attraverso il flag CY. Il bit di ordine basso è caricato con il flag CY e il flag CY è caricato con il valore ricevuto dal bit di ordine più alto. **Solo il flag CY è coinvolto.**

0	0	0	1	0	1	1	1

Formato esadecimale

RAL 17

Cicli: 1
Stati: 4
Flag: CY

Tutti i mnemonici sono registrati ® Intel Corporation 1976

RAR (Ruotare a destra attraverso il riporto)

$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$

$(A_7) \leftarrow (CY)$

Il contenuto dell'accumulatore viene fatto ruotare a destra di una posizione attraverso il flag CY. Il bit di ordine più alto è caricato con il flag CY e quest'ultimo è caricato con il valore ricevuto dal bit di ordine basso. **È coinvolto solo il flag CY.**

0	0	0	1	1	1	1	1

Formato esadecimale

RAR 1F

Cicli: 1
Stati: 4
Flag: CY

CMC (Complementare il riporto)

$(CY) \leftarrow \overline{(CY)}$

Viene complementato il flag CY. **Nessun altro flag viene coinvolto.**

0	0	1	1	1	1	1	1

Formato esadecimale

CMC 3F

Cicli: 1
Stati: 4
Flag: CY

CMA (Complementare l'accumulatore)

$(A) \leftarrow \overline{(A)}$

Viene complementato il contenuto dell'accumulatore (i bit zero diventano 1, i bit uno diventano 0). **Non è coinvolto nessun flag.**

0	0	1	0	1	1	1	1

Formato esadecimale

CMA 2F

Cicli: 1
Stati: 4
Flag: nessuno

STC (Settare il riporto)

$(CY) \leftarrow 1$

Il flag CY è posto ad 1. **Non è coinvolto nessun altro flag.**

0	0	1	1	0	1	1	1

Formato esadecimale

STC 37

Cicli: 1
Stati: 4
Flag: CY

Tutti i mnemonici sono registrati ® Intel Corporation 1976

Gruppo di salto

Questo gruppo di istruzioni altera il normale flusso sequenziale di un programma.

I flag di condizione non sono alterati da nessuna istruzione di questo gruppo.

I due tipi di istruzioni di salto sono incondizionate e condizionate. I trasferimenti incondizionati eseguono semplicemente l'operazione specifica sul registro PC, il programma counter. I trasferimenti condizionati esaminano lo stato di uno dei quattro flag del processore - Zero, Sign, Parity o Carry - per decidere se l'operazione di salto deve essere eseguita o no. Le condizioni che possono essere specificate sono le seguenti:

CONDIZIONE	CCC
NZ— non zero (Z = 0)	000
Z— zero (Z = 1)	001
NC— non c'è riporto (CY = 0)	010
C— c'è riporto (CY = 1)	011
PO— parità dispari (P = 0)	100
PE— parità pari (P = 1)	101
P— più (S = 0)	110
M— meno (S = 1)	111

JMP addr (Salto)

(PC) ← (byte 3) (byte 2)

Il controllo è traferito all'istruzione il cui indirizzo è specificato nei byte 2 e 3 di questa istruzione.

1	1	0	0	0	0	1	1
indirizzo (ordine basso)							
indirizzo (ordine alto)							

Formato esadecimale

JMP addr C3

Cicli: 3
Stati: 10
Indirizzamento: immediato
Flag: nessuno

Jcondition addr (Salto condizionato)

Se (CCC),

(PC) ← (byte 3) (byte 2)

Se la condizione specifica è vera, il controllo è trasferito all'istruzione il cui indirizzo è specificato nei byte 3 e 2 dell'istruzione in corso; in caso contrario, il controllo continua sequenzialmente.

1	1	C	C	C	0	1	0
indirizzo (ordine basso)							
indirizzo (ordine alto)							

Formato esadecimale

JNZ	addr	C2	(salto se non zero)
JZ	addr	CA	(salto se zero)
JNC	addr	D2	(salto se non c'è riporto)
JC	addr	DA	(salto se c'è riporto)
JPO	addr	E2	(salto se parità dispari)
JPE	addr	EA	(salto se parità pari)
JP	addr	F2	(salto se positivo)
JM	addr	FA	(salto se negativo)

Cicli: 2 / 3
Stati: 7 / 10
Indirizzamento: immediato
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

CALL addr (Chiamata)

((SP) — 1) ← (PCH)
((SP) — 2) ← (PCL)
(SP) ← (SP) — 2
(PC) ← (byte 3) (byte 2)

Gli otto bit di ordine alto dell'indirizzo dell'istruzione successiva sono posti nella posizione di memoria il cui indirizzo è il contenuto del registro SP meno 1. Gli otto bit di ordine basso dell'indirizzo dell'istruzione successiva sono posti nella posizione di memoria il cui indirizzo è contenuto del registro SP meno due. Il contenuto del registro SP è decrementato di 2. Il controllo è trasferito all'istruzione il cui indirizzo è specificato nei byte 3 e 2 dell'istruzione in corso.

1	1	0	0	1	1	0	1
indirizzo (ordine basso)							
indirizzo (ordine alto)							

Formato esadecimale

CALL addr CD

Cicli: 5
Stati: 18
Indirizzamento: immediato / reg. indiretto
Flag: nessuno

Ccondition addr (Chiamata condizionata)

Se (CCC),
((SP) — 1) ← (PCH)
((SP) — 2) ← (PCL)
(SP) ← (SP) — 2
(PC) ← (byte 3) (byte 2)

Se la condizione specifica è vera, vengono eseguite le azioni specifiche nell'istruzione di chiamata (vedi sopra), altrimenti il controllo continua sequenzialmente.

1	1	C	C	C	1	0	0
indirizzo (ordine basso)							
indirizzo (ordine alto)							

Formato esadecimale

CNZ	addr	C4	(chiamata se non zero)
CZ	addr	CC	(chiamata se zero)
CNC	addr	D4	(chiamata se non c'è riporto)
CC	addr	DC	(chiamata se c'è riporto)
CPO	addr	E4	(chiamata se parità dispari)
CPE	addr	EC	(chiamata se parità pari)
CP	addr	F4	(chiamata se positivo)
CM	addr	FC	(chiamata se negativo)

Cicli: 2 / 5
Stati: 9 / 18
Indirizzamento: immediato / reg. indiretto
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

RET (Ritorno)

(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (SP) + 2)

Il contenuto della posizione di memoria il cui indirizzo è specificato dal registro SP è posto negli otto bit di ordine basso del registro PC. Il contenuto della posizione di memoria il cui indirizzo è il contenuto del registro SP più uno è posto negli otto bit di ordine più alto del registro PC. Il contenuto del registro SP è incrementato di 2.

1	1	0	0	1	0	0	1

Formato esadecimale

RET C9

Cicli: 3
Stati: 10
Indirizzamento: registro indiretto
Flag: nessuno

Rcondition (Ritorno incondizionato)

Se (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2

Se la condizione specifica è vera, vengono eseguite le azioni specifiche nell'istruzione RET (vedi sopra); altrimenti, il controllo continua sequenzialmente.

1	1	C	C	C	0	0	0

Formato esadecimale

RNZ	C0	(ritorno se non zero)
RZ	C8	(ritorno se zero)
RNC	D0	(ritorno se non c'è riporto)
RC	D8	(ritorno se c'è riporto)
RPO	E0	(ritorno se parità dispari)
RPE	E8	(ritorno se parità pari)
RP	F0	(ritorno se positivo)
RM	F8	(ritorno se negativo)

Cicli: 1/3
Stati: 6/2
Indirizzamento: registro indiretto
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

RST n (Restart)

((SP) — 1) ← (PCH)
((SP) — 2) ← (PCL)
(SP) ← (SP) — 2
(PC) ← 8 * (NNN)

Gli otto bit di ordine più alto dell'indirizzo dell'istruzione successiva vengono spostati nella posizione di memoria il cui indirizzo è il contenuto del registro SP meno uno. Gli otto bit di ordine basso dell'indirizzo dell'istruzione successiva vengono spostati nella posizione di memoria il cui indirizzo è il contenuto del registro SP meno due. Il contenuto del registro SP è decrementato di due. Il controllo è trasferito all'istruzione il cui indirizzo è otto volte il contenuto di NNN.

1	1	N	N	N	1	1	1

Formato esadecimale

RST 0 C7
RST 1 CF
RST 2 D7
RST 3 DF
RST 4 E7
RST 5 EF
RST 6 F7
RST 7 FF

Cicli: 3
Stati: 12
Indirizzamento: registro indiretto
Flag: nessuno

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

Program Counter dopo l'esecuzione di una Restart

PCHL (Saltare ad H ed L indirettamente spostare H ed L nel PC)

(PCH) ← (H)
(PCL) ← (L)

Il contenuto del registro H è caricato negli otto bit di ordine più alto del registro PC. Il contenuto del registro L è caricato negli otto bit di ordine basso del registro PC.

1	1	1	0	1	0	0	1

Formato esadecimale

PCHL E9

Cicli: 1
Stati: 6
Indirizzamento: registro
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

Gruppo stack, I / O, controllo macchina

Questo gruppo di istruzioni esegue I / O, manipola lo stack e altera i flag di controllo interni.

Salvo diversa indicazione, **i flag di condizione non vengono coinvolti da nessuna istruzione di questo gruppo.**

PAROLA DEI FLAG

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

x: non definito

PUSH rp (Inserire nello stack)

((SP) — 1) ← (rh)
((SP) — 2) ← (rl)
(SP) ← (SP) — 2

Il contenuto del registro di ordine più alto della coppia di registri rp è spostato nella posizione di memoria il cui indirizzo è il contenuto del registro SP meno uno. Il contenuto del registro di ordine basso della coppia di registri rp è spostato nella posizione di memoria il cui indirizzo è il contenuto del registro SP meno due. Il contenuto del registro SP è decrementato di 2. **Nota: la coppia di registri rp = SP può non essere specificata.**

1	1	R	P	0	1	0	1

Formato esadecimale

PUSH	B	C5	(inserisci la coppia di registri B e C)
PUSH	D	D5	(inserisci la coppia di registri D e E)
PUSH	H	E5	(inserisci la coppia di registri H e L)

Cicli: 3
Stati: 12
Indirizzamento: registro indiretto
Flag: nessuno

PUSH PSW (Inserire la parola di stato nello stack)

((SP) — 1) ← (A)
$((SP) - 2)_0 \leftarrow (CY)$, $((SP) - 2)_1 \leftarrow X$
$((SP) - 2)_2 \leftarrow (P)$, $((SP) - 2)_3 \leftarrow X$
$((SP) - 2)_4 \leftarrow (AC)$, $((SP) - 2)_5 \leftarrow X$
$((SP) - 2)_6 \leftarrow (Z)$, $((SP) - 2)_7 \leftarrow (S)$
(SP) ← (SP) — 2 X: Non definito

Il contenuto del registro A è spostato nella posizione di memoria il cui indirizzo è il registro SP meno uno: Il contenuto dei flag viene raccolto in una parola di stato (Processor Status Word, PSW) e tale parola è spostata nella posizione di memoria il cui indirizzo è il contenuto del registro SP meno due. Il contenuto del registro SP è decrementato di due.

1	1	1	1	0	1	0	1

Formato esadecimale

PUSH PSW F5

Cicli: 3
Stati: 12
Indirizzamento: registro indiretto
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

POP rp (Estrarre dallo stack)

(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

Il contenuto della posizione di memoria, il cui indirizzo è specificato dal contenuto del registro SP, è spostato nel registro di ordine basso della coppia di registri rp. Il contenuto della posizione di memoria, il cui indirizzo è il contenuto del registro SP più uno, è spostato nel registro di ordine più alto della coppia di registri rp. Il contenuto del registro SP è incrementato di 2. **Nota: la coppia di registri rp = SP può non essere specificata.**

1	1	R	P	0	0	0	1

Formato esadecimale

POP	B	C1	(estrai la coppia di registri B e C)
POP	D	D1	(estrai la coppia di registri D e E)
POP	H	E1	(estrai la coppia di registri H e L)

Cicli: 3
Stati: 10
Indirizzamento: registro indiretto
Flag: nessuno

POP PSW (Estrarre la parola di stato)

(CY) ← $((SP))_0$
(P) ← $((SP))_2$
(AC) ← $((SP))_4$
(Z) ← $((SP))_6$
(S) ← $((SP))_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

Il contenuto della posizione di memoria il cui indirizzo è specificato dal contenuto del registro SP viene usato per ripristinare i flag di condizione. Il contenuto della posizione di memoria il cui indirizzo è il contenuto del registro SP più uno, è caricato nel registro A. Il contenuto del registro SP è incrementato di 2.

1	1	1	1	0	0	0	1

Formato esadecimale

POP PSW F1

Cicli: 3
Stati: 10
Indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

Tutti i mnemonici sono registrati ® Intel Corporation 1976

XTHL (Scambiare la parte superiore dello stack con H ed L)

(L) ← ((SP))
(H) ↔ ((SP) + 1)

Il contenuto del registro L è scambiato con il contenuto della posizione di memoria il cui indirizzo è specificato dal contenuto del registro SP. Il contenuto del registro H è scambiato con il contenuto della posizione di memoria il cui indirizzo è il contenuto del registro SP più uno.

1	1	1	0	0	0	1	1

Formato esadecimale

XTHL E3

Cicli: 5
Stati: 16
Indirizzamento: registro indiretto
Flag: nessuno

IN port (ingresso)

(A) ← (dato)

I dati posti dalla porta specificata sul bus di dati bidirezionale a otto bit sono caricati nel registro A.

1	1	0	1	1	0	1	1
porta							

Formato esadecimale

IN port DB

Cicli: 3
Stati: 10
Indirizzamento: diretto
Flag: nessuno

SPHL (Spostare HL in SP)

(SP) ← (H) (L)

Il contenuto dei registri H ed L (16 bit) è spostato nel registro SP.

1	1	1	1	1	0	0	1

Formato esadecimale

SPHL F9

Cicli: 1
Stati: 6
Indirizzamento: registro
Flag: nessuno

OUT port (uscita)

(dato) ← (A)

Il contenuto del registro A è posto sul bus di dati bidirezionale a 8 bit per essere caricato nella porta specifica.

1	1	0	1	0	0	1	1
porta							

Formato esadecimale

OUT port D3

Cicli: 3
Stati: 10
Indirizzamento: diretto
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

EI (Abilitare le interruzioni)
Il meccanismo dell'interruzione è abilitato **dopo l'esecuzione dell'istruzione successiva.**

1	1	1	1	1	0	1	1

Formato esadecimale

EI FB

Cicli: 1
Stati: 4
Flag: nessuno

Nota: Durante l'istruzione EI le interruzioni non sono riconosciute.

HLT (Alt)
Il processore viene fermato. I registri ed i flag non sono alterati.

0	1	1	1	0	1	1	0

Formato esadecimale

HLT 76

Cicli: 1
Stati: 5
Flag: nessuno

DI (Disabilitare le interruzioni)
Il meccanismo dell'interruzione è disabilitato **immediatamente dopo l'esecuzione della istruzione DI.**

1	1	1	1	0	0	1	1

Formato esadecimale

DI F3

Cicli: 1
Stati: 4
Flag: nessuno

Nota: Durante l'istruzione DI le interruzioni non sono riconosciute.

NOP (Nessuna operazione)
Non viene eseguita nessuna operazione. I registri ed i flag non sono alterati

0	0	0	0	0	0	0	0

Formato esadecimale

NOP 00

Cicli: 1
Stati: 4
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

RIM (Leggere la maschera delle interruzioni)

Dopo l'esecuzione dell'istruzione RIM, l'accumulatore si trova caricato con la maschera delle interruzioni Restart, con il flag Interrupt Enable (abilitazione delle interruzioni), con le indicazioni se si hanno delle interruzioni pendenti e con il contenuto della linea d'ingresso seriale (SID).

Quando, dopo un'interruzione TRAP, viene eseguita una prima volta l'istruzione RIM, il bit IE nell'accumulatore rispecchierà lo stato *precedente* dell'interrupt Enable, prima che si fosse verificato il TRAP. La prima RIM dopo un TRAP avrà perciò l'effetto di liberare il bit di stato IE, in modo che tutte le RIM successive possano disporre dello stato corrente dell'abilitazione delle interruzioni.

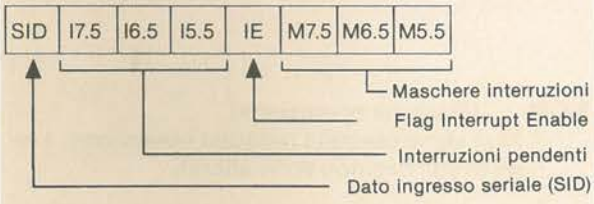
IMPORTANTE: Dopo un'interruzione TRAP, al fine di ripristinare lo stato corretto di Interrupt Enable, deve essere eseguita l'istruzione RIM.

0	0	1	0	0	0	0	0

Formato esadecimale

RIM 20

CONTENUTO DELL'ACCUMULATORE DOPO RIM



Cicli: 1
Stati: 4
Flag: nessuno

SIM (Predispone la maschera delle interruzioni)

Durante l'esecuzione dell'interruzione SIM, il contenuto dell'accumulatore viene utilizzato per programmare le maschere delle interruzioni Restart. Se il bit 3 è 1 (settato), i bit 0 - 2 settano / resettano il bit di maschera del registro maschera interruzioni per RST 5.5, 6.5, 7.5. Il bit 3 è un controllo «Abilitazione definizione maschera».

Portare a uno un bit della maschera vuol dire **disabilitare** l'interruzione corrispondente.

	Settata	Resettata
RST 5.5 MASK	se bit 0 = 1	se bit 0 = 0
RST 6.5 MASK	bit 1 = 1	bit 1 = 0
RST 7.5 MASK	bit 2 = 1	bit 2 = 0

Il flip flop di richiesta relativo a interno RST 7.5 (attivato dal fronte) verrà resettato se il bit 4 dell'accumulatore = 1, indipendentemente dal fatto che RST 7.5 sia mascherata o meno.

Un RESET hardware dell'8085A porterà a uno tutte le maschere RST e resetterà / disabiliterà tutte le interruzioni.

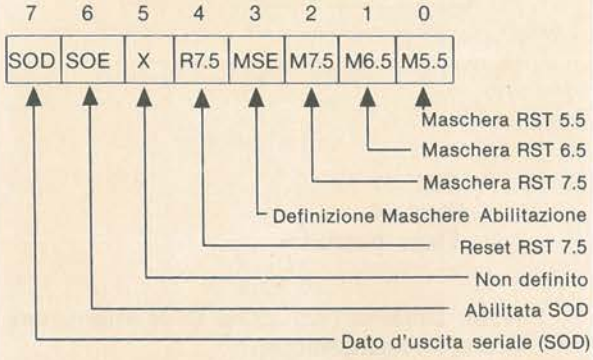
SIM serve anche a caricare il latch d'uscita di SOD, linea di uscita seriale. Il bit 7 dell'accumulatore viene caricato nel latch SOD se è posto a 1 il bit 6. Se il bit 6 è zero, il latch non verrà alterato. L'ingresso $\overline{\text{RESET IN}}$ pone a zero il latch SOD.

0	0	1	1	0	0	0	0

Formato esadecimale

SIM 30

CONTENUTO DELL'ACCUMULATORE PRIMA DI SIM



Cicli: 1
Stati: 4
Flag: nessuno

Tutti i mnemonici sono registrati ® Intel Corporation 1976

APPENDICE C

Tabelle delle firme

INTRODUZIONE

Questa appendice contiene le tabelle delle firme del μLab. Queste tabelle danno un elenco di tutti i piedini di tutti i circuiti integrati e, inoltre, delle linee dei bus degli indirizzi e dei dati.

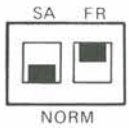
Nella maggior parte dei casi, per ogni piedino viene mostrata una firma; possono però esistere diversi modi per rappresentare la firma. Se il piedino è collegato direttamente a massa (Ground abbreviata GND) o a +5V, nella tabella invece della firma verrà indicato semplicemente GND o Vcc. La firma di Vcc è riportata all'inizio della tabella, mentre la firma di GND è sempre 0000. La tabella può anche riportare, come firme, i valori 0 o 1, che hanno lo stesso significato di GND e Vcc, con la sola differenza che il segnale è l'uscita di un dispositivo e non un collegamento diretto a GND o Vcc. Da ultimo la tabella può riportare una firma di Vcc o di GND seguita da una B; si vuole così dare una indicazione analoga a quella di 1 o 0, ma con la differenza che la luce sul puntale della sonda dell'analizzatore di firma dovrebbe lampeggiare. Così, ad esempio, nella Tabella C-2 la firma del piedino 3 di IC2 è 7A70B. Questo significa che, quando si verifica il fronte del clock, il segnale è sempre allo stesso livello logico (alto), ma che il segnale negli altri istanti è ad un livello differente. Per una spiegazione completa di come è possibile utilizzare le tabelle delle firme al fine di ricercare guasti nel μLab, fate riferimento alla Lezione 17.

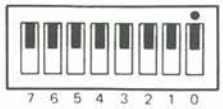
TABELLA C-1. FIRME IN FREE-RUN RELATIVE AGLI INDIRIZZI

5004A INTERRUTTORI	5036A COLLEGAMENTI
START ⎍	A15
STOP ⎍	A15
CLOCK ⎍	$\overline{\text{READ}}$

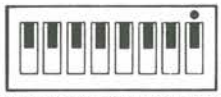
FIRMA DI V_{CC} :0001

5036A INTERRUTTORI





INTERRUTTORI D'INGRESSO



INTERRUTTORI DEL BUS

Linee dati		Linee indirizzi			
D0	X	A0	UUUU	A8	HC89
D1	X	A1	5555	A9	2H70
D2	X	A2	CCCC	A10	HPP0
D3	X	A3	7F7F	A11	1293
D4	X	A4	5H21	A12	HAP7
D5	X	A5	0AFA	A13	3C96
D6	X	A6	UPFH	A14	3827
D7	X	A7	52F8	A15	755P

IC1

GND	1	20	VCC
1293	2	19	0000
1293	3	18	755P
HPP0	4	17	755P
HPP0	5	16	3827
2H70	6	15	3827
2H70	7	14	3C96
HC89	8	13	3C96
HC89	9	12	HAP7
GND	10	11	HAP7

IC2

0000	1	20	VCC
UUUU	2	19	52F8
0001-B	3	18	0000-B
0001-B	4	17	0001-B
5555	5	16	UPFH
CCCC	6	15	0AFA
0001-B	7	14	0001-B
0001-B	8	13	0001-B
7F7F	9	12	5H21
GND	10	11	0001-B

IC3

X	1	40	VCC
X	2	39	0000
0000	3	38	0000
0000 or 0001	4	37	0001-B
0000	5	36	0001
0000	6	35	0001
0000	7	34	0000
0000	8	33	0001
0000	9	32	0001-B
0000	10	31	0001
0001	11	30	0000-B
0001-B	12	29	0001
0001-B	13	28	755P
0001-B	14	27	3827
0001-B	15	26	3C96
0001-B	16	25	HAP7
0001-B	17	24	1293
0001-B	18	23	HPP0
0000-B	19	22	2H70
GND	20	21	HC89

IC4

52F8	1	24	VCC
UPFH	2	23	HC89
0AFA	3	22	2H70
5H21	4	21	0001
7F7F	5	20	3PCF
CCCC	6	19	HPP0
5555	7	18	0000-B
UUUU	8	17	X
X	9	16	X
X	10	15	X
X	11	14	X
GND	12	13	X

IC5

UPFH	1	18	VCC
0AFA	2	17	52F8
5H21	3	16	HC89
7F7F	4	15	2H70
UUUU	5	14	X
5555	6	13	X
CCCC	7	12	X
84AF	8	11	X
GND	9	10	0001

IC6

UPFH	1	18	VCC
0AFA	2	17	52F8
5H21	3	16	HC89
7F7F	4	15	2H70
UUUU	5	14	X
5555	6	13	X
CCCC	7	12	X
84AF	8	11	X
GND	9	10	0001

IC7

1293	1	16	VCC
HAP7	2	15	3PCF
3C96	3	14	84AF
3827	4	13	960F
755P	5	12	4154
0001-B	6	11	UA87
1920	7	10	597C
GND	8	9	C34C

IC8

0001	1	16	VCC
0000	2	15	F770
0001	3	14	F771
GND	4	13	CCCC
UUUU	5	12	5555
8HUC	6	11	U6AH
8HUA	7	10	U6AF
GND	8	9	960F

X = Firma non significativa B = Lampeggiante, oppure firma di V_{CC}

TABELLA C-1. FIRME IN FREE-RUN RELATIVE AGLI INDIRIZZI (continuazione)

IC9

0001	1	14	VCC
0001	2	13	HC89
0000	3	12	2H70
0001	4	11	1883
0000-B	5	10	1883
0001-B	6	9	0000-B
GND	7	8	0001-B

IC10

0001-B	1	14	VCC
VCC	2	13	U6AF
0000	3	12	VCC
71U6	4	11	5555
2F8U	5	10	VCC
2F8P	6	9	71U6
GND	7	8	71U7

IC11

84AF	1	14	VCC
0000-B	2	13	0001
84AF	3	12	F771
8HUC	4	11	0001
0001-B	5	10	0000-B
0001-B	6	9	4154
GND	7	8	4154

IC12

0001-B	1	14	VCC
0000-B	2	13	0001
0000-B	3	12	0000
0001-B	4	11	0000
0001	5	10	0001
0000	6	9	0001
GND	7	8	0000

IC13

0000-B	1	20	VCC
0001	2	19	UA87
X	3	18	0001
0001	4	17	X
X	5	16	0001
0001	6	15	X
X	7	14	0001
0001	8	13	X
X	9	12	0001
GND	10	11	X

IC14

GND	1	20	VCC
X	2	19	GND
X	3	18	X
X	4	17	X
X	5	16	X
X	6	15	X
X	7	14	X
X	8	13	X
X	9	12	X
GND	10	11	X

IC15

VCC	1	20	VCC
0001	2	19	0001
X	3	18	X
X	4	17	X
0001	5	16	0001
0001	6	15	0001
X	7	14	X
X	8	13	X
0001	9	12	0001
GND	10	11	C34C

IC16

VCC	1	20	VCC
0001	2	19	0001
X	3	18	X
X	4	17	X
0001	5	16	0001
0001	6	15	0001
X	7	14	X
X	8	13	X
0001	9	12	0001
GND	10	11	1920

IC17

VCC	1	20	VCC
0001	2	19	0001
X	3	18	X
X	4	17	X
0001	5	16	0001
0001	6	15	0001
X	7	14	X
X	8	13	X
0001	9	12	0001
GND	10	11	597C

IC18

0000	1	20	VCC
0001	2	19	4154
0001	3	18	0001
0001-B	4	17	X
0000-B	5	16	0001
0000 or 0001	6	15	X
0000 or 0001	7	14	0001
0000 or 0001	8	13	X
0000 or 0001	9	12	0001
GND	10	11	X

IC19

0001	1	18	VCC
0001	2	17	X
0001	3	16	X
0001	4	15	X
0001	5	14	X
0001	6	13	X
0001	7	12	X
0001	8	11	X
X	9	10	X

IC20

X	1	14	0001
X	2	13	X
0001	3	12	0001
GND	4	11	VCC
0001	5	10	0001
X	6	9	X
X	7	8	0001

TABELLA C-2. FIRME IN FREE-RUN RELATIVE ALLA ROM

5004A INTERRUTTORI	5036A COLLEGAMENTI
START ⊓	$\overline{ROM}$
STOP ⊔	$\overline{ROM}$
CLOCK ⊔	$\overline{READ}$

FIRMA DI V_{cc} :7A70

5036A INTERRUTTORI

SA FR

NORM

FREERUN

NORM

INTERRUTTORI DEL BUS

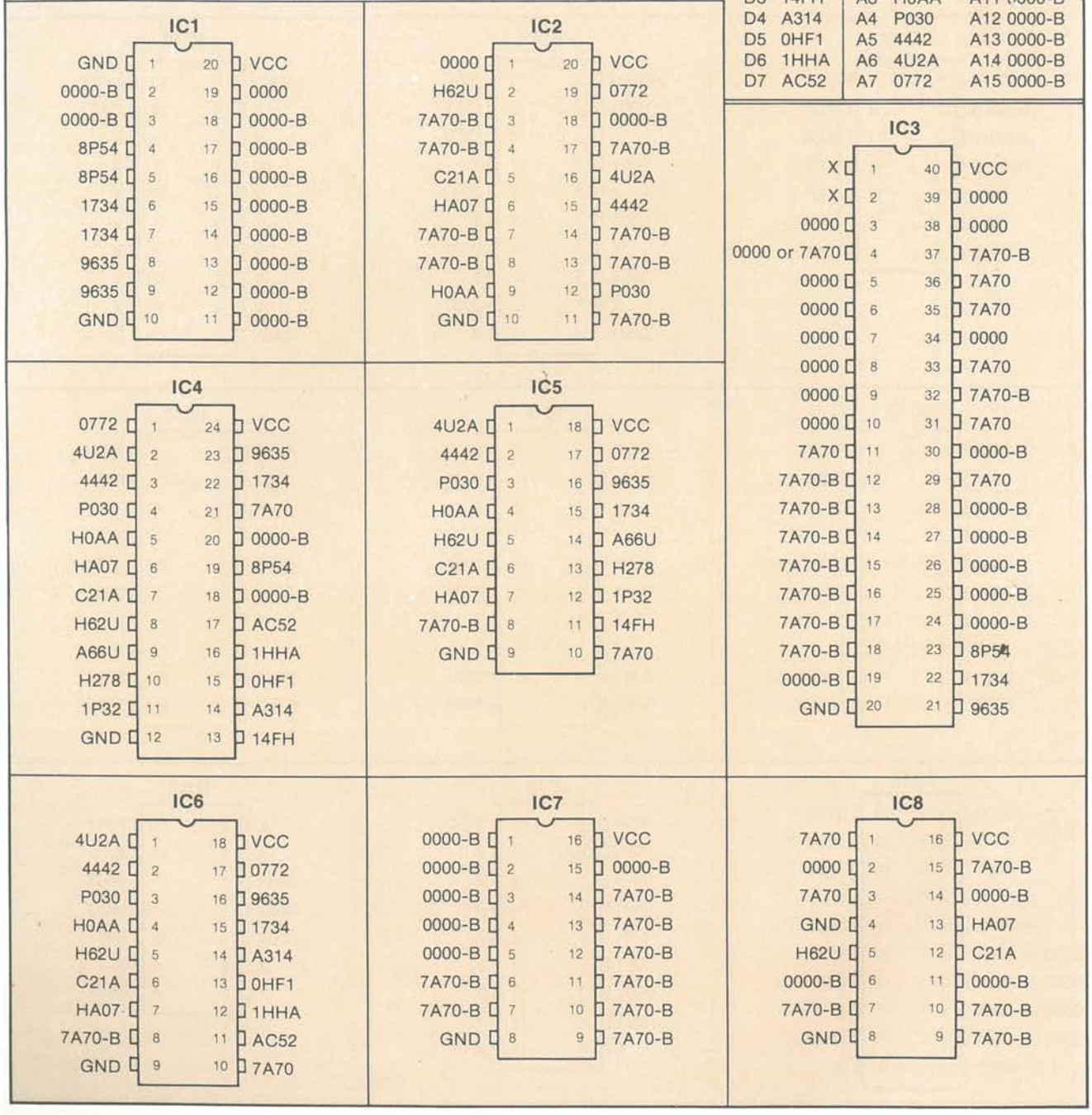
1 LOGICO

0 LOGICO

7 6 5 4 3 2 1 0

INTERRUTTORI D'INGRESSO

Linee dati		Linee indirizzi			
D0	A66U	A0	H62U	A8	9635
D1	H278	A1	C21A	A9	1734
D2	1P32	A2	HA07	A10	8P54
D3	14FH	A3	H0AA	A11	0000-B
D4	A314	A4	P030	A12	0000-B
D5	0HF1	A5	4442	A13	0000-B
D6	1HHA	A6	4U2A	A14	0000-B
D7	AC52	A7	0772	A15	0000-B



X = Firma non significativa B = Lampeggiante, oppure firma di V_{cc}

TABELLA C-2. FIRME IN FREE-RUN RELATIVE ALLA ROM (continuazione)

IC9

7A70	1	14	VCC
7A70	2	13	9635
0000	3	12	1734
7A70	4	11	P5AP
0000-B	5	10	P5AP
7A70-B	6	9	0000-B
GND	7	8	7A70-B

IC10

7A70-B	1	14	VCC
VCC	2	13	7A70-B
0000	3	12	VCC
7A70-B	4	11	C21A
0000-B	5	10	VCC
7A70-B	6	9	7A70-B
GND	7	8	0000-B

IC11

7A70-B	1	14	VCC
0000-B	2	13	7A70
7A70-B	3	12	0000-B
0000-B	4	11	7A70
7A70-B	5	10	0000-B
7A70-B	6	9	7A70-B
GND	7	8	7A70-B

IC12

7A70-B	1	14	VCC
0000-B	2	13	7A70
0000-B	3	12	0000
7A70-B	4	11	0000
7A70	5	10	7A70
0000	6	9	7A70
GND	7	8	0000

IC13

0000-B	1	20	VCC
7A70	2	19	7A70-B
A66U	3	18	7A70
7A70	4	17	AC52
H278	5	16	7A70
7A70	6	15	1HHA
1P32	7	14	7A70
7A70	8	13	0HF1
14FH	9	12	7A70
GND	10	11	A314

IC14

GND	1	20	VCC
A66U	2	19	GND
A66U	3	18	AC52
H278	4	17	AC52
H278	5	16	1HHA
1P32	6	15	1HHA
1P32	7	14	0HF1
14FH	8	13	0HF1
14FH	9	12	A314
GND	10	11	A314

IC15

VCC	1	20	VCC
7A70	2	19	7A70
A66U	3	18	AC52
H278	4	17	1HHA
7A70	5	16	7A70
7A70	6	15	7A70
1P32	7	14	0HF1
14FH	8	13	A314
7A70	9	12	7A70
GND	10	11	7A70-B

IC16

VCC	1	20	VCC
7A70	2	19	7A70
A66U	3	18	AC52
H278	4	17	1HHA
7A70	5	16	7A70
7A70	6	15	7A70
1P32	7	14	0HF1
14FH	8	13	A314
7A70	9	12	7A70
GND	10	11	7A70-B

IC17

VCC	1	20	VCC
7A70	2	19	7A70
A66U	3	18	AC52
H278	4	17	1HHA
7A70	5	16	7A70
7A70	6	15	7A70
1P32	7	14	0HF1
14FH	8	13	A314
7A70	9	12	7A70
GND	10	11	7A70-B

IC18

0000	1	20	VCC
7A70	2	19	7A70-B
7A70	3	18	7A70
7A70-B	4	17	14FH
0000-B	5	16	7A70
0000 or 7A70	6	15	1P32
0000 or 7A70	7	14	7A70
0000 or 7A70	8	13	H278
0000 or 7A70	9	12	7A70
GND	10	11	A66U

IC19

7A70	1	18	VCC
7A70	2	17	X
7A70	3	16	X
7A70	4	15	X
7A70	5	14	X
7A70	6	13	X
7A70	7	12	X
7A70	8	11	X
X	9	10	X

IC20

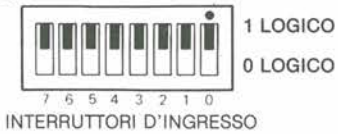
X	1	14	7A70
X	2	13	X
7A70	3	12	7A70
GND	4	11	VCC
7A70	5	10	7A70
X	6	9	X
X	7	8	7A70

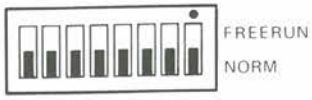
TABELLA C-3. FIRME IN S.A. DI SCRITTURA

5004A INTERRUTTORI	5036A COLLEGAMENTI
START ⎍	A15
STOP ⎍	A15
CLOCK ⎍	$\overline{\text{WRITE}}$

FIRMA DI V_{CC} :ICAU

5036A INTERRUTTORI





Linee dati		Linee indirizzi			
D0	937H	A0	UU3P	A8	9P91
D1	C072	A1	209C	A9	0765
D2	F5A5	A2	HHCO	A10	0000-B
D3	UA0P	A3	85A9	A11	1P3H
D4	51H6	A4	6F64	A12	1C32
D5	AU8U	A5	7A21	A13	7A26
D6	2PPH	A6	A2UC	A14	0000-B
D7	H011	A7	3483	A15	9678

IC1

GND	1	20	VCC
1P3H	2	19	0000
1P3H	3	18	9678
0000-B	4	17	9678
0000-B	5	16	0000-B
0765	6	15	0000-B
0765	7	14	7A26
9P91	8	13	7A26
9P91	9	12	1C32
GND	10	11	1C32

IC2

0000	1	20	VCC
UU3P	2	19	3483
937H	3	18	H011
C072	4	17	2PPH
209C	5	16	A2UC
HHC0	6	15	7A21
F5A5	7	14	AU8U
UA0P	8	13	51H6
85A9	9	12	6F64
GND	10	11	1CAU-B

IC3

X	1	40	VCC
X	2	39	0000
0000	3	38	0000
1CAU-B	4	37	ICAU-B
0000	5	36	1CAU
0000	6	35	1CAU
0000	7	34	FP05
0000	8	33	0000-B
0000	9	32	1CAU-B
0000	10	31	1CAU-B
1CAU	11	30	0000-B
937H	12	29	1CAU-B
C072	13	28	9678
F5A5	14	27	0000-B
UA0P	15	26	7A26
51H6	16	25	1C32
AU8U	17	24	1P3H
2PPH	18	23	0000-B
H011	19	22	0765
GND	20	21	9P91

IC4

3483	1	24	VCC
A2UC	2	23	9P91
7A21	3	22	0765
6F64	4	21	1CAU
85A9	5	20	1CAU-B
HHC0	6	19	0000-B
209C	7	18	1CAU-B
UU3P	8	17	H011
937H	9	16	2PPH
C072	10	15	AU8U
F5A5	11	14	51H6
GND	12	13	UA0P

IC5

A2UC	1	18	VCC
7A21	2	17	3483
6F64	3	16	9P91
85A9	4	15	0765
UU3P	5	14	937H
209C	6	13	C072
HHC0	7	12	F5A5
1FFA	8	11	UA0P
GND	9	10	0000-B

IC6

A2UC	1	18	VCC
7A21	2	17	3483
6F64	3	16	9P91
85A9	4	15	0765
UU3P	5	14	51H6
209C	6	13	AU8U
HHC0	7	12	2PPH
1FFA	8	11	H011
GND	9	10	0000-B

IC7

1P3H	1	16	VCC
1C32	2	15	1CAU-B
7A26	3	14	FP05
0000-B	4	13	39U4
9678	5	12	1CAU-B
1CAU-B	6	11	1CAU-B
9377	7	10	58P0
GND	8	9	AA1P

IC8

1CAU	1	16	VCC
0000	2	15	0000
1CAU	3	14	1CAU
GND	4	13	HHC0
UU3P	5	12	209C
0599	6	11	1CAU
1P36	7	10	0000
GND	8	9	39U4

X = Firma non significativa B = Lampeggiante, oppure firma di V_{CC}

TABELLA C-3. FIRME IN S.A. DI SCRITTURA (continuazione)

IC9

1CAU	1	14	VCC
1CAU	2	13	9P91
0000	3	12	0765
0000-B	4	11	1FFH
1CAU-B	5	10	1FFH
1CAU-B	6	9	1CAU-B
GND	7	8	0762

IC10

1CAU-B	1	14	VCC
VCC	2	13	0000
0000	3	12	VCC
0000	4	11	209C
1CAU	5	10	VCC
0000-B	6	9	0000
GND	7	8	1CAU

IC11

FP05	1	14	VCC
CCA8	2	13	1CAU
1FFA	3	12	1CAU
0599	4	11	1CAU
0762	5	10	1CAU-B
A007	6	9	1CAU-B
GND	7	8	1CAU-B

IC12

A007	1	14	VCC
CCA8	2	13	1CAU
0000-B	3	12	0000
1CAU-B	4	11	0000
1CAU	5	10	1CAU
0000	6	9	1CAU
GND	7	8	0000

IC13

1CAU-B	1	20	VCC
1CAU	2	19	1CAU-B
937H	3	18	1CAU
1CAU	4	17	H011
C072	5	16	ICAU
1CAU	6	15	2PPH
F5A5	7	14	1CAU
1CAU	8	13	AU8U
UA0P	9	12	1CAU
GND	10	11	51H6

IC14

GND	1	20	VCC
937H	2	19	GND
937H	3	18	H011
C072	4	17	H011
C072	5	16	2PPH
F5A5	6	15	2PPH
F5A5	7	14	AU8U
UA0P	8	13	AU8U
UA0P	9	12	51H6
GND	10	11	51H6

IC15

VCC	1	20	VCC
HA59	2	19	CCC4
937H	3	18	H011
C072	4	17	2PPH
PH2F	5	16	7769
7696	6	15	PPH2
F5A5	7	14	AU8U
UA0P	8	13	51H6
CC4C	9	12	HHA5
GND	10	11	AA1P

IC16

VCC	1	20	VCC
6PPH	2	19	22HH
937H	3	18	H011
C072	4	17	2PPH
C776	5	16	45CC
5CCC	6	15	8C77
F5A5	7	14	AU8U
UA0P	8	13	51H6
2HHH	9	12	16PP
GND	10	11	9377

IC17

VCC	1	20	VCC
65CU	2	19	C15H
937H	3	18	H011
C072	4	17	2PPH
8957	5	16	70PP
A5A1	6	15	1625
F5A5	7	14	AU8U
UA0P	8	13	51H6
6037	9	12	81H5
GND	10	11	58P0

IC18

0000	1	20	VCC
1CAU-B	2	19	1CAU-B
1CAU-B	3	18	ICAU
1CAU-B	4	17	UA0P
1CAU-B	5	16	1CAU
1CAU-B	6	15	F5A5
1CAU-B	7	14	1CAU
1CAU-B	8	13	C072
1CAU-B	9	12	1CAU
GND	10	11	937H

IC19

6PPH	1	18	VCC
C776	2	17	X
5CCC	3	16	X
2HHH	4	15	X
16PP	5	14	X
8C77	6	13	X
45CC	7	12	X
22HH	8	11	X
X	9	10	X

IC20

X	1	14	65CU
X	2	13	X
81H5	3	12	8957
GND	4	11	VCC
1625	5	10	A5A1
X	6	9	X
X	7	8	6037

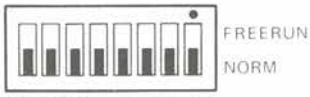
TABELLA C-4. FIRME IN S.A. DI LETTURA

5004A INTERRUTTORI	5036A COLLEGAMENTI
START ⎍	A15
STOP ⎍	A15
CLOCK ⎍	$\overline{\text{READ}}$

FIRMA DI V_{CC} :AU35

5036A INTERRUTTORI

INTERRUTTORI D'INGRESSO



INTERRUTTORI DEL BUS

Linee dati		Linee indirizzi			
D0	0122	A0	773F	A8	3U98
D1	A7FP	A1	4H2F	A9	3U98
D2	8863	A2	6087	A10	2UP0
D3	H3A4	A3	5HA7	A11	AAH3
D4	F616	A4	6757	A12	954C
D5	5C74	A5	HUC6	A13	52A9
D6	P165	A6	1C96	A14	0000-B
D7	36A9	A7	PAU5	A15	78AU

IC1

GND	1	20	VCC
AAH3	2	19	0000
AAH3	3	18	78AU
2UP0	4	17	78AU
2UP0	5	16	0000-B
3U98	6	15	0000-B
3U98	7	14	52A9
3U98	8	13	52A9
3U98	9	12	954C
GND	10	11	954C

IC2

0000	1	20	VCC
773F	2	19	PAU5
0122	3	18	36A9
A7FP	4	17	P165
4H2F	5	16	1C96
6087	6	15	HUC6
8863	7	14	5C74
H3A4	8	13	F616
5HA7	9	12	6757
GND	10	11	AU35-B

IC3

X	1	40	VCC
X	2	39	0000
0000	3	38	0000
13F3	4	37	AU35-B
0000	5	36	AU35
0000	6	35	AU35
0000	7	34	CU4H
0000	8	33	AU35-B
0000	9	32	AU35-B
0000	10	31	AU35-B
AU35	11	30	0000-B
0122	12	29	2F69
A7FP	13	28	78AU
8863	14	27	0000-B
H3A4	15	26	52A9
F616	16	25	954C
5C74	17	24	AAH3
P165	18	23	2UP0
36A9	19	22	3U98
GND	20	21	3U98

IC4

PAU5	1	24	VCC
1C96	2	23	3U98
HUC6	3	22	3U98
6757	4	21	AU35
5HA7	5	20	80H5
6087	6	19	2UP0
4H2F	7	18	0000-B
773F	8	17	36A9
0122	9	16	P165
A7FP	10	15	5C74
8863	11	14	F616
GND	12	13	H3A4

IC5

1C96	1	18	VCC
HUC6	2	17	PAU5
6757	3	16	3U98
5HA7	4	15	3U98
773F	5	14	0122
4H2F	6	13	A7FP
6087	7	12	8863
90AH	8	11	H3A4
GND	9	10	AU35-B

IC6

1C96	1	18	VCC
HUC6	2	17	PAU5
6757	3	16	3U98
5HA7	4	15	3U98
773F	5	14	F616
4H2F	6	13	5C74
6087	7	12	P165
90AH	8	11	36A9
GND	9	10	AU35-B

IC7

AAH3	1	16	VCC
954C	2	15	80H5
52A9	3	14	90AH
0000-B	4	13	AU35
78AU	5	12	3A7P
AU35-B	6	11	UH9F
AU35-B	7	10	AU35-B
GND	8	9	AU35-B

IC8

AU35	1	16	VCC
0000	2	15	0000
AU35	3	14	AU35
GND	4	13	6087
773F	5	12	4H2F
H443	6	11	AU35
7C76	7	10	0000
GND	8	9	AU35-B

X = Firma non significativa B = Lampeggiante, oppure firma di V_{CC}

TABELLA C-4. FIRME IN S.A. DI LETTURA (continuazione)

IC9

AU35	1	14	VCC
AU35	2	13	3U98
0000	3	12	3U98
AU35-B	4	11	90AH
0000-B	5	10	90AH
AU35-B	6	9	0000-B
GND	7	8	AU35-B

IC10

AU35-B	1	14	VCC
VCC	2	13	0000
0000	3	12	VCC
0000	4	11	4H2F
AU35	5	10	VCC
0000-B	6	9	0000
GND	7	8	AU35

IC11

90AH	1	14	VCC
0000-B	2	13	AU35
90AH	3	12	AU35
H443	4	11	AU35
AU35-B	5	10	0000-B
AU35-B	6	9	3A7P
GND	7	8	3A7P

IC12

AU35-B	1	14	VCC
0000-B	2	13	AU35
0000-B	3	12	0000
AU35-B	4	11	0000
AU35	5	10	AU35
0000	6	9	AU35
GND	7	8	0000

IC13

0000-B	1	20	VCC
AU35	2	19	UH9F
0122	3	18	AU35
AU35	4	17	36A9
A7FP	5	16	AU35
AU35	6	15	P165
8863	7	14	AU35
AU35	8	13	5C74
H3A4	9	12	AU35
GND	10	11	F616

IC14

GND	1	20	VCC
0122	2	19	GND
0122	3	18	36A9
A7FP	4	17	36A9
A7FP	5	16	P165
8863	6	15	P165
8863	7	14	5C74
H3A4	8	13	5C74
H3A4	9	12	F616
GND	10	11	F616

IC15

VCC	1	20	VCC
C202	2	19	0AF8
0122	3	18	36A9
A7FP	4	17	P165
H2F8	5	16	C214
674C	6	15	852F
8863	7	14	5C74
H3A4	8	13	F616
F19H	9	12	4C06
GND	10	11	AU35-B

IC16

VCC	1	20	VCC
6C01	2	19	C139
0122	3	18	36A9
A7FP	4	17	P165
91AF	5	16	4P53
CA46	6	15	94F9
8863	7	14	5C74
H3A4	8	13	F616
92P9	9	12	324C
GND	10	11	AU35-B

IC17

VCC	1	20	VCC
HF4A	2	19	4787
0122	3	18	36A9
A7FP	4	17	P165
7H76	5	16	882F
3011	6	15	P21H
8863	7	14	5C74
H3A4	8	13	F616
4923	9	12	H071
GND	10	11	AU35-B

IC18

0000	1	20	VCC
AU35-B	2	19	3A7P
AU35-B	3	18	AU35
AU35-B	4	17	H3A4
0000-B	5	16	AU35
13F3	6	15	8863
13F3	7	14	AU35
13F3	8	13	A7FP
13F3	9	12	AU35
GND	10	11	0122

IC19

6C01	1	18	VCC
91AF	2	17	X
CA46	3	16	X
92P9	4	15	X
324C	5	14	X
94F9	6	13	X
4P53	7	12	X
C139	8	11	X
X	9	10	X

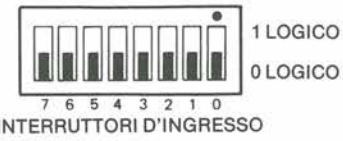
IC20

X	1	14	HF4A
X	2	13	X
H071	3	12	7H76
GND	4	11	VCC
P21H	5	10	3011
X	6	9	X
X	7	8	4923

TABELLA C-4. FIRME IN S.A. DI LETTURA (continuazione)

NOTA

Per il Test della Porta d'ingresso e per il test della tastiera predisponete tutti gli INTERRUTTORI D'INGRESSO verso il basso e verificate le seguenti firme per le linee dei dati:



Test porta d'ingresso

D0	538C
D1	U567
D2	HAFA
D3	810H
D4	94CU
D5	09HH
D6	C3FF
D7	6400

Test tastiera

D0	F6F0	quando è premuto il tasto 0, 1, 4, 7, A o d
D1	602F	quando è premuto il tasto 2, 5, 8, b o E
D2	4U81	quando è premuto il tasto 3, 6, 9, C o F
D3	1446	quando è premuto il tasto HDWR STEP

APPENDICE D

Lettura degli schemi elettrici

Per poter capire e controllare dei sistemi basati su microprocessore è assolutamente necessario che siate in grado di leggere gli schemi elettrici. Per una presentazione completa dei simboli logici fate riferimento a «HP's Practical Digital Electronics Course».

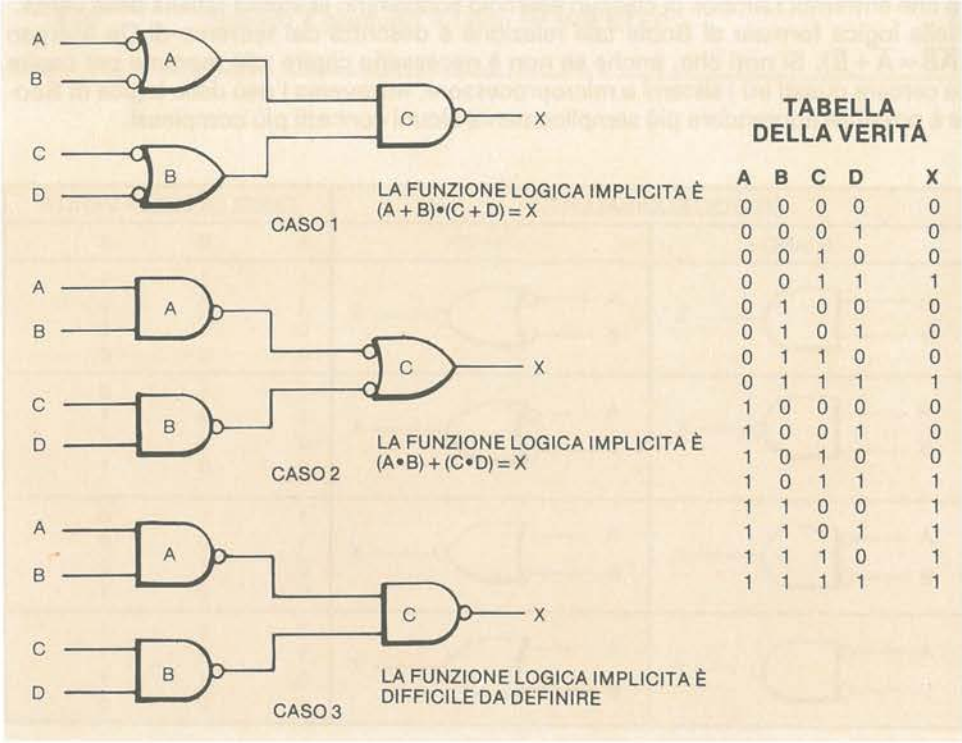
Parlando di simboli logici, un aspetto che sembra spesso ingenerare più confusione del necessario è quello che riguarda gli indicatori di livello logico basso (i piccoli cerchi). È possibile usare questi indicatori di livello logico basso in modo da associare allo stesso dispositivo fisico due differenti simboli logici. Per spiegare questo importante concetto, nella Tabella D-1 vengono mostrate le relazioni che esistono tra il dispositivo fondamentale AND e il dispositivo OR.
Confrontando come variano i simboli rispetto alle tabelle della verità, potete verificare che entrambi i simboli di ciascun esempio soddisfano la stessa tabella della verità.
Nella logica formale di Boole tale relazione è descritta dal teorema di De Morgan ($\overline{AB} = \overline{A} + \overline{B}$). Si noti che, anche se non è necessario capire tale teorema per capire (e cercare guasti in) i sistemi a microprocessore, attraverso l'uso della logica di Boole è possibile apprendere più semplicemente alcuni concetti più complessi.

SIMBOLI EQUIVALENTI		TABELLA DELLA VERITÁ		
(N)AND	(N)OR	A	B	X
		1 1 0 0	1 0 1 0	1 0 0 0
		1 1 0 0	1 0 1 0	0 1 1 1
		1 1 0 0	1 0 1 0	0 0 0 1
		1 1 0 0	1 0 1 0	1 1 1 0

Figura D.1 - Schemi logici funzionali

Leggendo degli schemi elettrici, potete notare che questi simboli sono talvolta utilizzati in modo erroneo. Questo è dovuto al fatto che la persona che ha disegnato lo schema non ha seguito le convenzioni che regolano l'uso di tali simboli. Lo scopo di avere due simboli diversi (per dispositivi identici) è quello di poter utilizzare, all'interno di un certo circuito, quel simbolo che rappresenta la funzione logica del dispositivo. Sfortunatamente, poiché sa come il dispositivo opera, il progettista non si preoccupa talvolta di fare distinzione tra identici dispositivi fisici usati in modi differenti. Ogni dispositivo viene così ad essere rappresentato con il simbolo indicato nelle specifiche date dal costruttore. Ad evitare tale problema, i dispositivi mostrati in questo corso sono disegnati in modo da indicare la funzione logica corretta all'interno del circuito.

Come esempio della differenza che si ha utilizzando una simbologia corretta, osserviamo la Figura D-1. Tutti i circuiti indicati sono identici e possono essere realizzati con dei dispositivi 7400. La scelta dei simboli è determinata da quanto volete trasmettere al lettore. Osservate le differenze che si hanno nei tre casi.
Nel caso 1, si vuole sicuramente indicare che sono dei segnali di livello basso ad attivare i dispositivi A e B e che l'uscita attiva del dispositivo C è un segnale di livello basso. Con queste informazioni diventa più facile l'operazione di verifica del circuito. Il caso 2 è simile ma lascia intendere che gli ingressi e le uscite del gruppo di dispositivi sono segnali attivi alti. Nel caso 3 invece è usato lo stesso simbolo logico per tutti e tre i dispositivi, con il risultato che diventa difficile definire esattamente come si vuole che il circuito operi. Viene creata un'ulteriore complicazione dal fatto che le uscite, che sembrerebbero attive basse, dei dispositivi A e B sono collegate agli ingressi, implicitamente attivi alti, del dispositivo C. Il punto essenziale da ricordare è che il circuito opera comunque nello stesso modo. Il solo motivo che porta ad utilizzare simboli differenti è che tali simboli aiutano l'utilizzatore a comprendere, all'interno di una ben precisa applicazione, la funzione dei circuiti. Sfortunatamente il numero dei circuiti disegnati facendo uso dei simboli standard che compaiono nelle specifiche del fabbricante è superiore a quello dei circuiti disegnati con i simboli funzionali. Ricordandovi le differenze che si hanno tra i due modi di disegnare schemi elettrici, vi sarà facile capire come un certo circuito funziona.



A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Tabella D-1. Relazioni tra i dispositivi (gate) fondamentali

APPENDICE E

Programmi dimostrativi e di utilità

Questa appendice spiega le procedure richieste per eseguire i programmi dimostrativi e di utilità. Nella Appendice F potrete trovare il listing completo di tali programmi.

PROGRAMMI DIMOSTRATIVI

Sono disponibili nove programmi dimostrativi, ognuno dei quali può essere eseguito caricando l'indirizzo di partenza e premendo RUN. Per comodità, nella Tabella E-1, sono stati riassunti i nomi e i rispettivi indirizzi di partenza di tutti i programmi. Quando avrete preso confidenza con tali programmi, le uniche informazioni che vi serviranno saranno proprio quelle contenute in tali Tabelle. Le diverse procedure sono però descritte anche in dettaglio, e di tali descrizioni potrete comunque fare uso.

Memoria	Indirizzo partenza	Descrizione
ECHO	04D7	Presenta sui LED d'uscita il dato degli interruttori d'ingresso.
ANDGT	04E0	Fa sì che gli interruttori di ingresso e la porta d'uscita operino come un dispositivo AND.
CONV	04F8	Controllo per il nastro trasportatore.
WTM	053E	Il microprocessore ben temperato genera da ROM delle note casuali.
SQRL	055A	Display di un registro con shift reazionato a "scoiattolo".
ORGAN	0599	Generazione di note tramite tastiera.
ROCT	05F9	Decollo missile.
STW	0662	Cronometro.
SNAKE	06C2	Ping Pong a serpente.

Tabella E-1. Programmi dimostrativi

Procedura ECHO (presenta sui LED di uscita il dato degli interruttori d'ingresso).

a. Premete il tasto RESET, se il display non visualizza già ⊔L Ab UP.
b. Premete il tasto FETCH ADRS e, utilizzando i tasti esadecimali, introducete il valore 04D7.
c. Premete RUN per far partire il programma.
d. Muovete gli interruttori d'ingresso (INPUT SWITCHES) impostando, a caso, delle combinazioni di 0 e di 1. Notate che i LED corrispondenti rispecchiano quanto impostato sugli interruttori di ingresso. Tenete ben presente che ogni LED si accende quando l'interruttore d'ingresso corrispondente è a 0 (LOGICO).
e. Per uscire dal programma premete RESET.

Procedura ANDGT (fa sì che gli interruttori d'ingresso e la porta d'uscita operino come un dispositivo AND):

a. Premete il tasto RESET, se il display non visualizza già ⊔L Ab
b. Premete il tasto FETCH ADRS e, utilizzando i tasti esadecimali, introducete il valore 04E0.
c. Premete RUN per far partire il programma.
d. Portate tutti gli interruttori d'ingresso (INPUT SWITCHES) ad 1 (LOGICO) e verificate che il LED d'uscita 0 è acceso.
e. Portate uno o più degli interruttori d'ingresso a 0 (LOGICO) e verificate che il LED d'uscita 0 si spegne.
f. Per uscire dal programma premete RESET.

Procedura CONV (controllore del nastro trasportatore discusso nella Lezione 3):

a. Premete il tasto RESET, se il display non visualizza già ⊔L Ab UP.
b. Premete il tasto FETCH ADRS e utilizzando i tasti esadecimali, introducete 04F8.
c. Premete RUN per far partire il programma. Il display visualizza 0, ad indicare il numero di ingranaggi che sono passati attraverso la fotocellula.
d. Premete il tasto 0. Così facendo simulate un impulso della fotocellula. Il display, di conseguenza, si modifica.
e. Ripetete il passo d finché il display non indichi che sono passati dieci ingranaggi. I LED d'uscita (OUTPUT LED) simulano a questo punto il movimento del nastro trasportatore di scatole e viene generato un suono.
f. Ripetete il passo e, verificando così il ripetersi del processo.
g. Per uscire dal programma premete RESET.

Procedura WTM (il «microprocessore ben temperato» genera da ROM note casuali):

a. Premete il tasto RESET, se il display non visualizza già ⊔L Ab UP.
b. Premete il tasto FETCH ADRS e utilizzando i tasti esadecimali introducete il valore 053E.
c. Premete RUN per far partire il programma.
d. Per uscire dal programma premete RESET.

Procedura SQRL (display di un registro con shift reazionato a «scoiattolo»):

a. Premete il tasto RESET, se il display non visualizza già ⊔L Ab UP.
b. Premete il tasto FETCH ADRS e, utilizzando i tasti esadecimali, introducete il valore 055A.
c. Premete RUN per far partire il programma.
d. Osservate, sul display ADDRESS/REGISTER (indirizzo registro), la configurazione che si sposta.

Nota

Questa configurazione è generata attraverso un algoritmo simile a quello utilizzato nell'analizzatore di firma per la compressione dei dati.

e. Per uscire dal programma, premete RESET.

Procedura ORGAN (genera le note indicate dalla tastiera):

a. Premete il tasto RESET, se il display non visualizza già ⊔L Ab UP.
b. Premete il tasto FETCH ADRS e, utilizzando i tasti esadecimali, introducete il valore 0599.
c. Premete RUN per far partire il programma.

Nota

I tasti esadecimali controllano ora la generazione delle diverse note. La nota di frequenza più bassa è controllata dal tasto 0, quella della frequenza più alta è controllata da F. Deve essere premuto un solo tasto per volta. La Tabella E-2 dà un elenco dei tasti e delle note corrispondenti.

	Nota	Tasti
	D	0
	E	1
	F	2
	A	3
	B	4
Centrale	C	5
	D	6
	E	7
	F	8
	G	9
	A	A
	B	b
	C	C
	D	d
	E	E
	F	F

Tabella E-2. Note musicali per il programma Organ

d. Premete una qualunque sequenza di tasti esadecimali. Se avete anche un po' di talento musicale, potete suonare qualche semplice melodia.
e. Per uscire dal programma premete RESET.

Procedura ROCT (decollo del missile)

a. Premete il tasto RESET, se il display non visualizza già uLAb UP.
b. Premete il tasto FETCH ADRS e, utilizzando i tasti esadecimali, introducete il valore 05F9.
c. Premete RUN per far partire il programma.
d. Osservate il «conto alla rovescia» sul display, seguito dal rumore e dagli effetti luminosi che simulano la partenza del razzo.
e. Per uscire dal programma premete RESET; per ripeterlo premete RUN.

Procedura STW (cronometro)

a. Premete il tasto RESET, se il display non visualizza già uLAb UP.
b. Premete il tasto FETCH ADRS e, utilizzando i tasti esadecimali, introducete il valore 0662.
c. Premete RUN per far partire il programma.
d. Osservate che il display visualizza 0 00 00.
e. Premete il tasto 3 e verificate che il display inizia a contare, incrementandosi ad ogni centesimo di secondo. Premete ancora il tasto 3 per fermare il temporizzatore. Il tasto 3 serve alternativamente per far partire e per fermare il temporizzatore. Il temporizzatore può contare fino a 9 minuti 59 secondi e 99 centesimi di secondo dopo di che riprenderà da capo.
f. Premete il tasto 0 per azzerare il temporizzatore.
g. Per uscire dal programma premete RESET.

Procedura SNAKE (ping-pong a serpente)

a. Premete il tasto RESET, se il display non visualizza già uLAb UP.
b. Premete il tasto FETCH ADRS e, utilizzando i tasti esadecimali, introducete il valore 06C2.
c. Premete RUN, per far partire il programma.

Nota

Le regole del gioco sono molto semplici. Il tasto DECR controlla la paletta di sinistra, mentre il tasto 3 controlla quella di destra. La paletta di sinistra (DECR) dà inizio al gioco servendo la piccola lineetta rossa in direzione dell'avversario (potete anche giocare da soli, contro voi stessi). Incominciamo ora a giocare: le altre regole vi verranno spiegate nel corso del gioco.

d. Il giocatore di sinistra deve premere il tasto DECR. Notate che la lineetta rossa incomincia a muoversi a zig-zag, come un «serpente», sul display.
e. Il giocatore della paletta di destra deve aspettare finché la lineetta rossa non raggiunge la posizione finale (angolo in alto a destra del display ADDRESS / REGISTER) e solo allora premere il tasto 3.

Nota

In questo gioco, l'abilità sta tutta nell'imparare a valutare correttamente quanto tempo la lineetta rossa si ferma, mentre si muove attraverso il display, sulle diverse posizioni. La valutazione di tale tempo è importante poiché la velocità del ritorno (osservate la Tabella E-3) è legata al tempo che la lineetta si trattiene nella posizione base (finale). L'obiettivo è quello di riuscire a premere il tasto durante il perido 3. Se la vostra risposta è corretta, il ritorno sarà più veloce e per il vostro avversario sarà più difficile rispondere. Un segnale acustico vi segnalerà la variazione della velocità di ritorno (più veloce o più lento).

Tabella E-3. Quando rispondere

f. Ora, chi si trova con la lineetta rossa deve battere premendo il tasto di controllo (DECR o 3).
g. Continuate a palleggiare e, scegliendo accuratamente il tempo, cercate di forzare un ritorno più veloce della lineetta rossa.
h. Quando uno dei due concorrenti sbaglia il tempo della risposta, l'avversario fa un punto. L'altoparlante emette un suono ad indicare che è stato segnato un punto. Il punteggio dei due giocatori è mostrato, rispecchiando le rispettive posizioni, sul display DATA. Ogni volta che fate un punto, il servizio passa a voi (lo conservate se già era vostro). La posizione della lineetta sul display indica chi deve servire. Il gioco termina al punteggio di 10. Avrà perciò vinto il gioco il primo dei due giocatori che vedrà comparire il carattere A sul suo lato del display. Quando il gioco è terminato, un giocatore è cioè arrivato ad A, i tasti di controllo diventano inefficaci.
i. Per ricominciare il gioco, premete il tasto 0. Così facendo la tabella con i punteggi viene azzerata e il giocatore con la paletta di sinistra può iniziare il gioco «servendo» la lineetta rossa al suo avversario.
j. Per uscire dal programma, premete RESET.

PROGRAMMI DI UTILITÀ

I programmi di utilità (utility programs) sono delle subroutine che potete utilizzare all'interno dei vostri programmi. La Tabella E-4 vi elenca i diversi programmi di utilità, dandovi l'indirizzo di partenza, i registri che la subroutine altera, i dati d'ingresso richiesti e una breve descrizione del programma stesso. Potete trovare alcuni esempi dell'utilizzo di questi programmi di utilità negli Esperimenti 14-1 e 14-3.

Nome programma	Registri alterati	Dati richiesti	Indirizzo di partenza	Commenti
BEEP	Tutti	Nessuno	0010*	Genera un bip di frequenza e durata fisse. Può essere chiamata anche introducendo un D7 all'interno del vostro programma. D7 è il comando di Restart 2 (RST 2)
BEEP 1	Tutti eccetto B	B contiene la frequenza	0012	Genera un bip di durata fissa e di frequenza definita dal registro B. Quando in B è contenuto un valore piccolo, il bip è ad alta frequenza (01 è il valore minimo permesso). Maggiore è il contenuto di B, minore sarà la frequenza del suono.
BEEP 2	Tutti eccetto B e D	B contiene la frequenza e D la durata	0447	Genera un bip la cui frequenza è definita dal registro B (Si veda BEEP 1). La durata del bip è definita dal registro D. Maggiore è il valore contenuto in D, maggiore sarà la durata del bip (01 è il valore minimo permesso).
KIND (Key input and Decode)	A	Ingressi tastiera	014B	Aggiorna il display nel mentre che effettua la scansione, l'antirimbalzo e la decodifica degli ingressi della tastiera. Il valore del tasto premuto (0-F) è posto nel registro A.
KPU (Key Pushed)	A, H e L	Ingressi tastiera	0185	Scandisce la tastiera e azzera il flag di zero se viene premuto un qualunque tasto.
SDS (Scan Display Segments)	Nessuno	Segmenti display	01C8	I dati relativi ai segmenti, memorizzati nelle locazioni RAM da 0BFA e 0BFF, sono inviati ai digit da 0 a 5 del display. Durante la scansione del display ogni cifra viene accesa per 1 ms (Si veda la Tabella E-5)

Tabella E-4 Routine di utilità

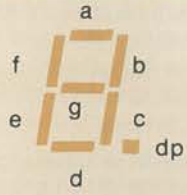
Nome programma	Registri alterati	Dati richiesti	Indirizzo di partenza	Commenti
DCD (Display Character Decoder)	Nessuno	Digit del display	01E9	Prende i codici (da 00 a IC) dei caratteri da visualizzare, memorizzati nelle locazioni RAM da 0BF0 a 0BF5, e li codifica in formato sette segmenti. Memorizza quindi questi ultimi nelle locazioni RAM da 0BFA a 0BFF e scandisce una volta il display. La decodifica viene effettuata utilizzando 0218 (Si vedano le tabelle E-6 e E-7).
STDM (Store Display Message)	Tutti	Coppia di registri DE	0018*	Prende il messaggio di sei caratteri, che inizia all'indirizzo contenuto nella coppia di registri DE e li memorizza nelle locazioni RAM del display UDSP, da 0BF0 a 0BF5. Il primo carattere del messaggio comparirà come ultima cifra a destra nel display. Questa routine è di solito seguita dalla subroutine DCD. La routine STDM può essere chiamata anche inserendo il codice DF nel vostro programma. DF è il comando di Restart 3. (RST 3).
DELA (Fixed Delay)	Nessuno	Nessuno	0429	Fa sì che venga generato un ritardo fisso di 1 ms.
DELB (Variable Delay)	Nessuno	Coppia di registri BC	0430	Fa sì che venga generato un ritardo variabile. Il ritardo è circa uguale al valore numerico contenuto nella coppia di registri BC moltiplicato per 1 ms.
RS1	Nessuno	Nessuno	0008*	Breakpoint software che fa ritornare al monitor. CF è il comando RST 1

* Può essere usato RST (si vedano i commenti descrittivi)

Tabella E-4 Routine di utilità

DDSP 0-5, Indirizzi 0BFA — 0BFF

Attivi bassi, un bit a zero accende il segmento corrispondente



byte dato

7	6	5	4	3	2	1	0
dp	g	f	e	d	c	b	a

Indirizzo RAM	Label	Digit decodificato del display (in codice 7 segmenti)
0BFA	DDSP0	0 (cifra più a destra)
0BFB	DDSP1	1
0BFC	DDSP2	2
0BFD	DDSP3	3
0BFE	DDSP4	4
0BFF	DDSP5	5 (cifra più a sinistra)

Tabella E-5. Cifre decodificate del display

Carattere Display	Codice carattere negli indirizzi 0BF0-0BF5
0	00
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
A	0A
b	0B
C	0C
d	0D
E	0E
F	0F
spazio	10
H	11
L	12
u	13
P	14
o	15
U	16
-	17
c	18
I	19
B	1A
r	1B
-	1C

Tabella E-6. Decodifica dei caratteri del display

UDSP 0-5, Indirizzi 0BF0 — 0BF5

I codici esadecimali dei caratteri vengono decodificati utilizzando la tabella Display Code Converter (DCC, cioè Convertitore codici display), che parte dall'indirizzo 0218 del listing del programma.

Indirizzi RAM	Label	Cifre non decodificate del display (in codice esadecimale)
0BF0	UDSP0	0 (cifra più a destra)
0BF1	UDSP1	1
0BF2	UDSP2	2
0BF3	UDSP3	3
0BF4	UDSP4	4
0BF5	UDSP5	5 (cifra più a sinistra)

Tabella E-7. Cifre non decodificate del display

APPENDICE F

Listing della ROM del Microprocessor lab

Questa appendice contiene il listing completo del programma monitor e dei programmi dimostrativi e di utilità che sono memorizzati nella ROM. Le procedure da seguire per utilizzare i programmi dimostrativi e di utilità sono presentate nell'Appendice E.

La Tabella F-1 contiene un elenco alfabetico dei simboli e delle label e gli indirizzi esadecimali che all'interno del programma vengono utilizzati. La Tabella F-2 è il listing del programma della ROM. Per ogni istruzione, la prima colonna contiene l'indirizzo esadecimale dell'istruzione. Notate che quando il codice di una istruzione richiede più di una locazione (cioè di un byte), l'elenco risulta discontinuo. Così, ad esempio, il codice JMP alla locazione 0005 richiede tre locazioni di memoria. In tale punto la sequenza degli indirizzi passa da 0005 a 0008, indicando così che il codice di JMP utilizza tre locazioni di memoria. La seconda colonna presenta il contenuto esadecimale di tale indirizzo. Ricordatevi che l'informazione di indirizzo è memorizzata in modo tale che il primo byte è quello meno significativo. Perciò, alla locazione 0005, il codice C34000 fa sì che venga eseguito un salto alla locazione 0040. Nella terza colonna vengono riportati i simboli, definiti dall'utente, che sono utilizzati come label. La quarta colonna contiene il codice mnemonico dell'istruzione. Nell'Appendice B potete trovare una descrizione completa dei codici e dei codici mnemonici. È importante ricordare che l'assembler dell'8085 si aspetta che tutti i dati numerici siano espressi in decimale. Quando viene usato un dato esadecimale, dopo il numero deve essere inserita una H. L'ultima colonna contiene dei brevi commenti fatti da chi ha scritto il programma. Dei commenti generali e delle descrizioni, che partono dall'inizio della quarta colonna e si estendono per la quinta colonna, sono invece posti all'inizio dei programmi, delle subroutine o dei moduli.

ALL	A 0281	ANDGT	A 04E0	BEEP	A 0010	BEEP1	A 0012	BEEP2	A 044
BEEP5	A 0464	BEEP6	A 046F	BEEP7	A 047E	BEEP8	A 0480	BEEP9	A 048
CDCRM	A 02B3	CDCRR	A 02CD	CFETA	A 02B8	CFETP	A 02BD	CFETR	A 020
CINSS	A 02A4	CODE	A 05BB	CONV	A 04F8	CRUN	A 02A9	CSTRM	A 02A
DCD	A 01E9	DCD1	A 01F3	DCD2	A 0210	DCRM	A 03AA	DCRR	A 03F
DDSP5	A 0BFF	DEL1	A 0431	DEL2	A 0434	DEL3	A 0436	DELA	A 042
DLY	A 05AE	DMT	A 0241	DPS	A 0319	DSP	A 0038	DTIME	A 005
ENDGM	A 06F7	FETA	A 0328	FETA1	A 032F	FETA2	A 034D	FETA3	A 036
FETA6	A 0393	FETA7	A 039C	FETAR	A 0329	FETCH	A 0247	FETR	A 02D
FETR3	A 02F1	FETR4	A 02FE	FETR5	A 0300	FETR6	A 0310	FLG	A 025
FSTTN	A 0001	HDSS	A 03F7	ICX	A 0287	IM	A 027D	INCR	A 001
KIND	A 014B	KIND1	A 014D	KIND2	A 0156	KIND3	A 0164	KIND4	A 017
KPU	A 0185	KPU1	A 0190	KRD	A 019A	KRD1	A 01A3	LEFT	A 00F
LOSTN	A 0028	LOUT	A 0030	LSM	A 07D4	MA	A 024D	MB	A 025
ME	A 0261	MERR	A 00DC	MERR1	A 00E5	MERR2	A 00ED	MH	A 026
MLP	A 0518	MOVE	A 0514	MSP	A 0BCE	NCTL	A 034A	NOKEY	A 05D
ORGAN	A 0599	ORGAN1	A 059B	PADL	A 07A2	PASS	A 0738	PASS1	A 073
PCL	A 0279	PLAY	A 075A	PLAY1	A 0767	PNTLS	A 00FF	PPER	A 00C
RAML2	A 0BD7	RBND	A 070C	RBND1	A 0727	READ	A 05C0	RESET	A 000
ROCM	A 065C	ROCT	A 05F9	ROCT1	A 0601	ROCT2	A 0603	ROCT3	A 061
RS	A 0BD6	RS1	A 0008	RS4	A 0020	RS4C	A 0AF0	RS5	A 002
RS5C	A 0AF3	RS6	A 0030	RS65	A 0034	RS65C	A 0AFC	RS6C	A 0AF
RSM	A 07E2	RSRV	A 0719	RSTGM	A 0799	RUN	A 03C2	RUN1	A 03C
SATL2	A 04A1	SATL3	A 04A7	SATL4	A 04AF	SAVA	A 0BE7	SAVE	A 0BE
SAVPC	A 0BDC	SAVSH	A 0BDF	SAVSL	A 0BDE	SCAN	A 0028	SDM	A 023
SDS1	A 01D0	SERV	A 06C8	SERV1	A 06EB	SERV2	A 0700	SHIFT	A 05C
SNAKE	A 06C2	SPEED	A 0B00	SPH	A 026D	SPKR	A 05B5	SPL	A 027
SQRL2	A 057B	SQRL3	A 0586	STDM	A 0018	STRM	A 03AF	STRR	A 041
STRT2	A 0064	STRT3	A 0073	STRT4	A 008A	STRT5	A 009C	STRT6	A 00A
STW1	A 0669	STW2	A 0693	STW3	A 069B	STW4	A 06A4	STW5	A 06A
STWM	A 06BC	TABLE	A 05E2	TIME	A 008C	TIME1	A 07B0	TIMER	A 07A
TRAP	A 0024	TRIL	A 07B9	TRIL1	A 07BF	TRIL2	A 07CD	TRP	A 00F
TRP3	A 0111	TRP4	A 0115	TRP5	A 011B	TRP6	A 0124	TRP7	A 014
UDKY	A 0BE8	UDSP0	A 0BF0	UDSP1	A 0BF1	UDSP2	A 0BF2	UDSP3	A 0BF
UR	A 0AEF	USP	A 0BB0	WTM	A 053E	WTM1	A 0541	XLOOP	A 07F

Tabella F-1. Simboli utilizzati

```
Hex
Adrs Contents  Label   Instruction          Comments

0000                   ORG  0
0000 2608      RESET:  MVI  H,6        ;PAGE ADRS OF WRITE TEST RAM
0002 7E                MOV  A,M        ;TEST RAM CELL DATA
0003 2F                CMA             ;COMPLEMENT IT
0004 77                MOV  M,A        ;STORE IT BACK IN RAM
0005 C34000            JMP  STRT       ;TO CONTINUE
                       ;
                       ;RS1 IS MAIN ENTRY POINT TO MONITOR
                       ;
0008                   ORG  8
0008 22D30B    RS1:    SHLD TSAVH      ;SAVE USER HL CONTENTS IN RAM
000B D310              OUT  10h        ;UNPROTECT RAM
000D C3F300            JMP  TRF        ;CONTINUE
                       ;
                       ;BEEP PRODUCES A FIXED TONE FREQ + DURATION
                       ;
0010                   ORG  10H
0010 0606      BEEP:   MVI  B,FREQ     ;DEFAULT FREQUENCY
0012 1604      BEEP1:  MVI  D,DURA     ;DEFAULT DURATION
0014 C34704            JMP  BEEP2      ;CONTINUE
0017 00                NOP
                       ;
                       ;STDM STORES DISP MESSAGE AT DE ADRS IN UDSP RAM
                       ;
0018                   ORG  18h
0018 C5        STDM:   PUSH B
0019 21F00B            LXI  H,LDSP0    ;ADRS OF UNDECODED DISPLAY DIGIT 0
001C C33502            JMP  SDM        ;CONTINUE
001F 00                NOP
0020                   ORG  20H
0020 C3F00A RS4:       JMP  RS4C       ;USER ROUTINE
0023 00                NOP
0024                   ORG  24h
0024 C30800 TRAP:      JMP  RS1        ;SAV USER REGS+RETURN TO MONITOR
0027 00                NOP
0028                   ORG  28h
0028 C3F30A RS5:       JMP  RS5C       ;USER ROUTINE
002B 00                NOP
002C                   ORG  2Ch
002C C3F60A RS55:      JMP  RS55C      ;USER ROUTINE
002F 00                NOP
0030                   ORG  30h
0030 C3F90A RS6:       JMP  RS6C       ;USER ROUTINE
0033 00                NOP
0034                   ORG  34h
0034 C3FC0A RS65:      JMP  RS65C      ;USER INTERRUPT KEY ROUTINE
0037 00                NOP
0038                   ORG  38h
0038 C3A600 RS7:       JMP  STRT6      ;RETURN TO MONITOR
003B 00                NOP
003C                   ORG  3Ch
003C D7        RS75:   RST  2          ;BEEP
003D C38D04            JMP  SATL1      ;SA TEST LOOP
                       ;
                       ;POWER-UP SELF TEST AND INITIALIZE
                       ;
0040                   ORG  40H
0040 BE        STRT:   CMP  M          ;SEE IF DATA STORED IN RAM
0041 C2C800            JNZ  PPER       ;IF RAM WAS PROTECTED (RUN MODE)
0044 31CE0B            LXI  SP,MSP     ;INITIALIZE MONITOR SP
0047 AF                XRA  A          ;CLEAR A
0048 67                MOV  H,A        ;CLEAR H      FIRST ADRS
0049 6F                MOV  L,A        ;CLEAR L      OF ROM
004A D330              OUT  LOUT       ;TURN ON OUTPUT LEDS
                       ;
                       ;ROM SELF TEST
                       ;
004C 86        STRT1:  ADD  M          ;ADD ROM DATA TO A
004D 23                INX  H          ;POINT TO NEXT ROM ADRS
004E 4F                MOV  C,A        ;SAVE A RESIDUE IN C
004F 3E08              MVI  A,08H      ;LAST ADRS OF ROM+1 (MSBYTE)
0051 BC                CMP  H          ;COMPARE IT TO H
0052 79                MOV  A,C        ;RESTORE RESIDUE TO A
0053 C24C00            JNZ  STRT1      ;IF LAST ROM ADRS NOT 0800
0056 2B                DCX  H          ;POINT HL TO 07FF CHECKSUM VALUE
0057 96                SUB  M          ;SUBTRACT CHECKSUM FOR A RESIDUE
0058 BE                CMP  M          ;COMPARE RESIDUE TO CHECKSUM
0059 0604              MVI  B,04       ;IC 4 (ROM) MESSAGE
005B C2E500            JNZ  MERR1      ;IF NOT A MATCH
                       ;
                       ;RAM SELF TEST
                       ;
005E AF                XRA  A          ;CLEAR A
005F 210008            LXI  H,0800H    ;1ST RAM ADDR
0062 0603              MVI  B,3        ;ADD CONSTANT
0064 77        STRT2:  MOV  M,A        ;STORE DATA IN RAM
0065 80                ADD  B          ;ADD 3 TO A
0066 23                INX  H          ;POINT TO NEXT RAM ADRS
0067 4F                MOV  C,A        ;SAVE A
0068 7C                MOV  A,H        ;GET MSBYTE ADRS
0069 FE0C              CPI  0CH        ;LAST RAM ADRS+1
006B 79                MOV  A,C        ;RESTORE A
006C C26400            JNZ  STRT2      ;IF NOT LAST RAM ADRS
006F AF                XRA  A          ;CLEAR A
0070 210008            LXI  H,0800H    ;1ST RAM ADRS
0073 BE        STRT3:  CMP  M          ;DID DATA GET STORED IN RAM?
0074 C2DC00            JNZ  MERR       ;IF DATA NOT SAME
0077 2F                CMA             ;
0078 77                MOV  M,A        ;STORE COMPLEMENT BACK IN RAM
0079 BE                CMP  M          ;DID IT STORE?
007A C2DC00            JNZ  MERR       ;IF NOT
007D 2F                CMA             ;UNCOMPLEMENT A
007E 80                ADD  B          ;ADD 3 TO A
007F 23                INX  H          ;NEXT RAM ADDR
0080 4F                MOV  C,A        ;SAVE A
0081 7C                MOV  A,H        ;GET MSBYTE ADRS
0082 FE0C              CPI  0CH        ;LAST RAM ADRS+1
0084 79                MOV  A,C        ;RESTORE A
```

```
Hex
Adrs Contents  Label   Instruction          Comments

0085 C27300            JNZ  STRT3      ;TO CHECK NEXT RAM LOCATION
                       ;
                       ;DISPLAY TEST
                       ;
0088 0600              MVI  B,0        ;CLEAR LOOP COUNTER
008A 118102 STRT4:     LXI  D,FLL      ;DISPLAY MESSAGE POINTER  ALL SEGS
008D DF                RST  3          ;GET MESSAGE
008E CDE901            CALL DCI        ;UPDATE DISPLAY
0091 05                DCR  B          ;DECR LOOP COUNTER
0092 C28A00            JNZ  STRT4      ;IF NOT DONE
                       ;
                       ;CLEAR RAM (STORE 00 IN ALL LOCATIONS)
                       ;
0095 0600              MVI  B,0        ;CLEAR B
0097 3E0C              MVI  A,0CH      ;MSBYTE ADRS OF TOP OF RAM+1
0099 210008            LXI  H,0800H    ;1ST ADRS OF RAM
009C 70        STRT5:  MOV  M,B        ;CLEAR RAM LOCATION
009D 23                INX  H          ;POINT TO NEXT LOCATION
009E BC                CMP  H          ;TO LAST RAM ADRS+1
009F C29C00            JNZ  STRT5      ;IF NOT DONE CLEARING RAM
00A2 3EFF              MVI  A,0FFH     ;SET A TO ALL ONES
00A4 D330              OUT  LOUT       ;TURN OFF OUTPUT LEDS
00A6 D7        STRT6:  RST  2          ;SIGNAL START-UP DONE
                       ;
                       ;INITIALIZE REGISTERS
                       ;
00A7 21B00B            LXI  H,LSP      ;USER SP DEFAULT VALUE
00AA 22DE0B            SHLD SAVSL      ;STORE IT IN RAM
00AD 21D60B            LXI  H,RS       ;RUN STATUS WORD ADRS
00B0 3600              MVI  M,0        ;SET STATUS TO MONITOR
00B2 3E0B              MVI  A,DIM      ;DEFAULT INTERRUPT MASK
00B4 32DB0B            STA  SAVIM      ;STORE IT IN RAM
00B7 210000 STRT7:     LXI  H,FC       ;DEFAULT PC
00BA 22DC0B            SHLD SAVPC      ;STORE IT IN RAM
00BD 3EFF              MVI  A,0FFH     ;RST7 INSTR CODE
00BF 32FF0A            STA  TPF        ;STORE IT IN TOP OF PROTECTED RAM
00C2 32EF0A            STA  UP         ;STORE IT IN UPPER RAM BELOW LINKS
00C5 C31101            JMP  TRF3       ;JUMP TO MONITOR
                       ;
                       ;PUSH-POP ERROR ROUTINE
                       ;
00C8 31CE0B PPER:      LXI  SP,MSP     ;SET MONITOR SP
00CB 210000            LXI  H,FC       ;DEFAULT PC
00CE 22DC0B            SHLD SAVPC      ;STORE IT IN RAM
00D1 D7                RST  2          ;SIGNAL AN ERROR
00D2 AF                XRA  A          ;CLEAR A
00D3 32B60B            STA  RS         ;SET RUN STATUS TO MONITOR
00D6 118D02            LXI  D,FPM      ;PUSH-POP ERROR MESSAGE ADRS
00D9 C31501            JMP  TRF4
                       ;
                       ;MEMORY ERROR SORT
                       ;
00DC 0606      MERR:   MVI  B,6        ;IC6 RAM FAIL MESSAGE
00DE AE                XRA  M          ;GET DIFFERENCE INTO A
00DF E60F              ANI  0FH        ;TEST 4LSB'S OF IC6
00E1 CAE500            JZ   MERR1      ;IF PROBLEM IN IC6
00E4 05                DCR  B          ;SET B TO IC5
00E5 118702 MERR1:     LXI  D,ICX      ;ICX MESSAGE ADRS
00E8 DF                RST  3          ;GET MESSAGE
00E9 21F20B            LXI  H,LDSP2    ;ADRS OF ICX IN UDSP RAM
00EC 70                MOV  M,B        ;STORE IC NUMBER IN UDSP2
00ED CDE901 MERR2:     CALL DCI        ;DISPLAY MESSAGE
00F0 C3ED00            JMP  MERR2      ;LOOP MESSAGE
                       ;
                       ;RS1 IS MAIN ENTRY POINT TO MONITOR
                       ;
00F3 210000 TRP:       LXI  H,0        ;CLEAR H,L
00F6 D2FA00            JNC  TRP1       ;NO USER CARRY WILL BYPASS DCXH
00F9 2B                DCX  H          ;SET H,L TO FF IF CARRY PRESENT
00FA 39        TRP1:   DAD  SP         ;GET SP VALUE INTO HL, RESTORE CY
00FB D2FF00            JNC  TRP2       ;IF JNC TO TRP1 OCCURED
00FE 23                INX  H          ;IF DCXH OCCURED BECAUSE OF CARRY
00FF 22D10B TRP2:      SHLD TSAVSP     ;SAVE USER SP IN RAM
0102 31D10B            LXI  SP,TSAVSP+1;TSAVA+1 ADRS
0105 F5                PUSH PSW        ;SAVE PSW AND A IN RAM
0106 21D60B            LXI  H,RS       ;RUN STATUS ADRS
0109 AF                XRA  A          ;CLEAR A
010A BE                CMP  M          ;FOR RUN STATUS=MONITOR
010B 32F60B            STA  UDSP6      ;CLEAR DATA MODIFY FLAG
010E C22401            JNZ  TRP6       ;IF CAME FROM USER PROGRAM
0111 114102 TRP3:      LXI  D,DMT      ;ULAB MESSAGE ADRS
0114 FB                EI
0115 3E0B      TRP4:   MVI  A,DIM      ;DEFAULT INTERRUPT MASK
0117 30                SIM             ;SET IT TO ENABLE RST7.5 ONLY
0118 D310              OUT  10h        ;UNPROTECT RAM
011A DF                RST  3          ;GET MESSAGE
011B CD4B01 TRP5:      CALL KIND       ;INPUT KEYS
011E CDEB02            CALL CFETA      ;LOOK FOR ACCEPTABLE KEYS
0121 C31B01            JMP  TRP5       ;TRY AGAIN
0124 77        TRP6:   MOV  M,A        ;STORE 0 IN RS TO SET MONITOR
                       ;
                       ;SAVES REGISTERS
                       ;
0125 F1                POP  PSW        ;FROM TSAVPSW IN RAM
0126 E1                POP  H          ;GETS USER SP VALUE FROM RAM
0127 23                INX  H          ;USER SP
0128 23                INX  H          ;USER SP
0129 22DE0B            SHLD SAVSL      ;SAVE SP IN RAM
012C 2B                DCX  H          ;USER SP
012D 2B                DCX  H          ;USER SP
012E F9                SPHL            ;RESTORE SP
012F E1                POP  H          ;GET RETURN ADRS TO USER PROGRAM
0130 22DC0B            SHLD SAVPC      ;STORE IT IN RAM
0133 31E80B            LXI  SP,0BE8H   ;ADRS OF SAVA+1
0136 2AD30B            LHLD TSAVH      ;RESTORE H,L
0139 F5                PUSH PSW        ;INTO SAVPSW
013A C5                PUSH B          ;INTO SAVB
013B D5                PUSH D          ;INTO SAVD
```

Tabella F-2. Listing della ROM

```
Hex
Adrs Contents  Label   Instruction          Comments

013C E5                PUSH H          ;INTO SAVM
013D 20                RIM             ;GET IM
013E 32DB0B            STA  SAVIM      ;STORE IT IN RAM
0141 31DC0B  TRP7:     LXI  SP,SAVPC   ;POINT IT TO USER SP
0144 C1                POP  B          ;AND POP IT IN BC
0145 31CE0B            LXI  SP,MSP     ;RESTORE MONTIOR SP
0148 C36403            JMP  FETA3
                       ;
                       ;KEY INPUT AND DECODE
                       ;
014B D5      KIND:     PUSH D
014C E5                PUSH H
014D CDE901  KIND1:    CALL DCD        ;UPDATE DISPLAY AND WAIT
0150 CD8501            CALL KPU        ;CHK FOR PUSHED KEY
0153 C24D01            JNZ  KIND1      ;IF KEY STILL PUSHED
0156 CDE901  KIND2:    CALL DCD        ;UPDATE DISP AND WAIT
0159 CD8501            CALL KPU        ;CHK FOR PUSHED KEY
015C CA5601            JZ   KIND2      ;IF KEY NOT PUSHED
015F 21E80B            LXI  H,UDKY     ;ADRS OF FIRST KEY ROW SCAN
0162 16FF              MVI  D,0FFH     ;LOAD ROW COUNTER TO 0-1
0164 7E      KIND3:    MOV  A,M        ;GET ROW N KEY DATA
0165 FEF7              CPI  0F7H       ;IS IT THE HDWR STEP*KEY?
0167 CA8201            JZ   KIND5      ;YES JUMPS
016A 2F                CMA             ;INVERT KEY DATA
016B 2C                INR  L          ;NEXT ROW
016C 14                INR  D          ;NEXT TABLE BLOCK
016D A7                ANA  A          ;TEST ROW N FOR 0
016E CA6401            JZ   KIND3      ;JUMP IF KEY NOT PUSHED
0171 FE04              CPI  4          ;SEE IF D3=1
0173 C27701            JNZ  KIND4      ;IF SO
0176 3D                DCR  A          ;ELSE SET A=3
0177 82      KIND4:    ADD  D          ;ADD 3X THE ROW N TO
0178 82                ADD  D          ;GET THE TABLE OFFSET
0179 82                ADD  D
017A 5F                MOV  E,A        ;STORE TABLE INDEX
017B 1600              MVI  D,0        ;CLEAR MS BYTE OF DE
017D 21AF01            LXI  H,KIT-1    ;ADRS OF KEY CODE TABLE
0180 19                DAD  D          ;ADD INDEX TO TABLE ADRS
0181 7E                MOV  A,M        ;PUT KEY CODE IN A
0182 E1      KIND5:    POP  H
0183 D1                POP  D
0184 C9                RET
                       ;
                       ;DETERMINES IF ANY KEY IS PUSHED
                       ;
0185 C5      KPU:      PUSH B
0186 CD9A01            CALL KRD        ;READ THE KEYBOARD
0189 0608              MVI  B,8        ;SET THE LOOP COUNTER
018B 21E80B            LXI  H,UDKY     ;ADDRESS OF UNDECODED KEY SCAN
018E 3EFF              MVI  A,0FFH     ;UNPUSHED KEY CODE
0190 A6      KPU1:     ANA  M          ;LET ANY PUSHED KEY CODE CHANGE A
0191 2C                INR  L          ;NEXT RAM KEY ROW
0192 05                DCR  B          ;LOOP COUNTER
0193 C29001            JNZ  KPL1       ;IF ALL KEY ROWS NOT ANDED WITH A
0196 FEFF              CPI  0FFH       ;SET FLAG IF ALL KEYS NOT PUSHED
0198 C1                POP  B
0199 C9                RET
                       ;
                       ;READS KEYS AND STORES THEM IN RAM (UDKY)
                       ;
019A 21E80B  KRD:      LXI  H,UDKY     ;ADRS OF UNDECODED KEY SCAN
019D 3EFF              MVI  A,0FFH     ;BLANK DISPLAY CODE
019F D330              OUT  DSP        ;CLEAR DISPLAY
01A1 3D                DCR  A          ;SET A TO 1111 1110 SCAN POINTER
01A2 37                STC             ;TO PRESET END OF SCAN LOOP FLAG
01A3 D328    KRD1:     OUT  SCAN       ;SCAN ONE KEY ROW
01A5 47                MOV  B,A        ;SAVE SCAN POINTER
01A6 DB18              IN   KEY        ;INPUT A KEY ROW
01A8 77                MOV  M,A        ;STORE IT IN RAM
01A9 7B                MOV  A,E        ;RESTORE SCAN POINTER
01AA 2C                INR  L          ;POINT TO NEXT RAM ADRS
01AB 17                RAL             ;MOVE SCAN POINTER TO NEXT KEY ROW
01AC DAA301            JC   KRD1       ;IF LAST KEY ROW NOT SCANNED
01AF C9                RET
                       ;
                       ;KEY INPUT DECODE TABLE (HDWR STEP IS F7)
                       ;
01B0 86      KIT:      DB   86H        ;INSTR STEP    KEY CODE
01B1 85                DB   85H        ;FETCH PC      KEY CODE
01B2 00                DB   0          ;(UNDEFINED)   KEY CODE
01B3 84                DB   84H        ;RUN           KEY CODE
01B4 80                DB   80H        ;FETCH REG     KEY CODE
01B5 82                DB   82H        ;FETCH ADRS    KEY CODE
01B6 00                DB   0          ;    0         KEY CODE
01B7 83                DB   83H        ;STORE/INCR    KEY CODE
01B8 81                DB   81H        ;DECR          KEY CODE
01B9 01                DB   1          ;    1         KEY CODE
01BA 02                DB   2          ;    2         KEY CODE
01BB 03                DB   3          ;    3         KEY CODE
01BC 04                DB   4          ;    4         KEY CODE
01BD 05                DB   5          ;    5         KEY CODE
01BE 06                DB   6          ;    6         KEY CODE
01BF 07                DB   7          ;    7         KEY CODE
01C0 08                DB   8          ;    8         KEY CODE
01C1 09                DB   9          ;    9         KEY CODE
01C2 0A                DB   0AH        ;    A         KEY CODE
01C3 0B                DB   0BH        ;    B         KEY CODE
01C4 0C                DB   0CH        ;    C         KEY CODE
01C5 0D                DB   0DH        ;    D         KEY CODE
01C6 0E                DB   0EH        ;    E         KEY CODE
01C7 0F                DB   0FH        ;    F         KEY CODE
                       ;
                       ;SCAN DISPLAY SEGMENTS
                       ;
01C8 F5      SDS:      PUSH PSW
01C9 E5                PUSH H
01CA C5                PUSH B
01CB 21FF0B            LXI  H,DDSP5    ;ADRS OF DECODED DISP DIGIT 5
01CE 0620              MVI  B,20H      ;DISP DIGIT 5 SCAN POINTER

Hex
Adrs Contents  Label   Instruction          Comments

01D0 AF      SDS1:     XRA  A          ;CLEAR A
01D1 D328              OUT  SCAN       ;TURN DISP DIGIT OFF
01D3 7E                MOV  A,M        ;GET SEGMENT DATA
01D4 D330              OUT  DSP        ;STORE IT IN DSP LATCH
01D6 78                MOV  A,E        ;GET SCAN DIGIT POINTER
01D7 D328              OUT  SCAN       ;TURN ON DISP DIGIT
01D9 CD2904            CALL DELA       ;STRETCH DIGIT IMS
01DC 2D                DCR  L          ;ADRS OF NEXT DIGIT IN RAM
01DD 1F                RAR             ;NEXT SCAN DIGIT POINTER
01DE 47                MOV  B,A        ;SAVE IT IN B
01DF D2D001            JNC  SDS1       ;IF NOT LAST SCANED DIGIT (LSD)
01E2 2F                CMA             ;SET A=FF BLANK DISP CODE
01E3 D330              OUT  DSP        ;TURN OFF DSP DIGITS
01E5 C1                POP  B
01E6 E1                POP  H
01E7 F1                POP  PSW
01E8 C9                RET
                       ;
                       ;DISPLAY CHARACTER DECODER
                       ;
01E9 F5      DCD:      PUSH PSW
01EA C5                PUSH B
01EB D5                PUSH D
01EC E5                PUSH H
01ED 01FA0B            LXI  B,DDSP0    ;DECODED DIGIT FIRST ADRS
01F0 11F00B            LXI  D,UDSP0    ;UNDECODED DIGIT FIRST ADRS
01F3 211802  DCD1:     LXI  H,DCC      ;DISPLAY CODE CONVERTER TABLE ADRS
01F6 1A                LDAX D          ;GET UNDECODED DATA FOR OFFSET
01F7 D5                PUSH D          ;SAVE ITS ADRS
01F8 5F                MOV  E,F        ;TABLE OFFSET VALUE TO E
01F9 1600              MVI  D,0        ;CLEAR D
01FB 19                DAD  D          ;ADD OFFSET VALUE TO DCC ADRS
01FC 7E                MOV  A,M        ;GED DECODED DATA FROM TABLE ADRS
01FD 02                STAX B          ;STORE IT IN DECODED RAM
01FE D1                POP  D          ;RESTORE UDSPX ADRS
01FF 1C                INR  E          ;POINT TO NEXT UPSP ADRS
0200 0C                INR  C          ;POINT TO NEXT DDSP ADRS
0201 C2F301            JNZ  DCD1       ;IF NOT LAST DIGIT
0204 21FA0B            LXI  H,DDSP0    ;DECODED DIGIT 0
0207 1A                LDAX D          ;UDSP6 DATA MODIFY FLAG
0208 A7                ANA  A          ;CHECK FOR SET FLAG
0209 CA1002            JZ   DCD2       ;IF DATA NOT BEING MODIFIED
020C 7E                MOV  A,M        ;GET DDSP0 DATA
020D E67F              ANI  7FH        ;SET ITS DECIMAL POINT DISP BIT
020F 77                MOV  M,A        ;STORE IT IN DDSP0
0210 E1      DCD2:     POP  H          ;
0211 D1                POP  D          ;
0212 C1                POP  B          ;
0213 F1                POP  PSW        ;
0214 CDC801            CALL SDS        ;UPDATE DISPLAY
0217 C9                RET             ;
                       ;
                       ;DISPLAY CODE CONVERTER TABLE
                       ;
0218 C0      DCC:      DB   0C0H       ;0
0219 F9                DB   0F9H       ;1
021A A4                DB   0A4H       ;2
021B B0                DB   0B0H       ;3
021C 99                DB   099H       ;4
021D 92                DB   092H       ;5
021E 82                DB   082H       ;6
021F F8                DB   0F8H       ;7
0220 80                DB   080H       ;8
0221 90                DB   090H       ;9
0222 88                DB   088H       ;A
0223 83                DB   083H       ;B
0224 C6                DB   0C6H       ;C
0225 A1                DB   0A1H       ;D
0226 86                DB   086H       ;E
0227 8E                DB   08EH       ;F
0228 FF                DB   0FFH       ;BLANK
0229 89                DB   089H       ;H
022A C7                DB   0C7H       ;L
022B E3                DB   0E3H       ;U SMALL
022C 8C                DB   08CH       ;P
022D A3                DB   0A3H       ;O
022E C1                DB   0C1H       ;U LARGE
022F F7                DB   0F7H       ;_
0230 A7                DB   0A7H       ;C SMALL
0231 CF                DB   0CFH       ;1 LEFT
0232 00                DB   00H        ;ALL SEGS
0233 AF                DB   0AFH       ;R SMALL
0234 BF                DB   0BFH       ;-
                       ;
                       ;STDM STORES DISP MESSAGE AT DE ADRS IN UDSP RAM
                       ;
0235 0606    SDM:      MVI  B,6        ;LOOP COUNTER FOR 6 DISPLAY DIGITS
0237 1A      SDM1:     LDAX D          ;DISPLAY CHARACTER
0238 77                MOV  M,A        ;STORE IT IN UPSPX IN RAM
0239 2C                INR  L          ;NEXT UDSP ADRS
023A 13                INX  D          ;NEXT MESSAGE TABLE ADRS
023B 05                DCR  B          ;LOOP COUNTER
023C C23702            JNZ  SDM1       ;IF NOT LAST DIGIT
023F C1                POP  B          ;
0240 C9                RET
                       ;
                       ;DISPLAY MESSAGE TABLES
                       ;
0241 14      DMT:      DB   14H        ;P
0242 16                DB   16H        ;U
0243 0B                DB   0BH        ;B
0244 0A                DB   0AH        ;A
0245 12                DB   12H        ;L
0246 13                DB   13H        ;U SMALL
0247 10      FETCH:    DB   10H        ;SP
0248 10                DB   10H        ;SP
0249 17                DB   17H        ;_
024A 17                DB   17H        ;_
024B 17                DB   17H        ;_
```

Tabella F-2. Listing della ROM (continuazione)

```
Hex
Adrs Contents  Label   Instruction          Comments

024C 17                DB   17H        ;_
024D 0A        MA:     DB   0AH        ;A
024E 10                DB   10H        ;SP
024F 10                DB   10H        ;SP
0250 10                DB   10H        ;SP
0251 12        FLG:    DB   12H        ;L
0252 0F                DB   0FH        ;F
0253 10                DB   10H        ;SP
0254 10                DB   10H        ;SP
0255 0B        MB:     DB   0BH        ;B
0256 10                DB   10H        ;SP
0257 10                DB   10H        ;SP
0258 10                DB   10H        ;SP
0259 0C        MC:     DB   0CH        ;C
025A 10                DB   10H        ;SP
025B 10                DB   10H        ;SP
025C 10                DB   10H        ;SP
025D 0D        MD:     DB   0DH        ;D
025E 10                DB   10H        ;SP
025F 10                DB   10H        ;SP
0260 10                DB   10H        ;SP
0261 0E        ME:     DB   0EH        ;E
0262 10                DB   10H        ;SP
0263 10                DB   10H        ;SP
0264 10                DB   10H        ;SP
0265 11        MH:     DB   11H        ;H
0266 10                DB   10H        ;SP
0267 10                DB   10H        ;SP
0268 10                DB   10H        ;SP
0269 12        ML:     DB   12H        ;L
026A 10                DB   10H        ;SP
026B 10                DB   10H        ;SP
026C 10                DB   10H        ;SP
026D 11        SPH:    DB   11H        ;H
026E 14                DB   14H        ;P
026F 05                DB   05H        ;S
0270 10                DB   10H        ;SP
0271 12        SPL:    DB   12H        ;L
0272 14                DB   14H        ;P
0273 05                DB   05H        ;S
0274 10                DB   10H        ;SP
0275 11        PCH:    DB   11H        ;H
0276 0C                DB   0CH        ;C
0277 14                DB   14H        ;P
0278 10                DB   10H        ;SP
0279 12        PCL:    DB   12H        ;L
027A 0C                DB   0CH        ;C
027B 14                DB   14H        ;P
027C 10                DB   10H        ;SP
027D 19        IM:     DB   19H        ;I
027E 10                DB   10H        ;SP
027F 10                DB   10H        ;SP
0280 10                DB   10H        ;SP
0281 1A        ALL:    DB   1AH        ;ALL
0282 1A                DB   1AH        ;ALL
0283 1A                DB   1AH        ;ALL
0284 1A                DB   1AH        ;ALL
0285 1A                DB   1AH        ;ALL
0286 1A                DB   1AH        ;ALL
0287 10        ICX:    DB   10H        ;SP
0288 10                DB   10H        ;SP
0289 10                DB   10H        ;SP
028A 0C                DB   0CH        ;C
028B 01                DB   01H        ;I
028C 10                DB   10H        ;SP
028D 1B        PPM:    DB   1BH        ;R SMALL
028E 0E                DB   0EH        ;E
028F 14                DB   14H        ;P
0290 05                DB   05H        ;S
0291 10        BLNKM:  DB   10H        ;SP  BLANK CHARACTER CODE
0292 10                DB   10H        ;SP
0293 10                DB   10H        ;SP
0294 10                DB   10H        ;SP
0295 10                DB   10H        ;SP
0296 10                DB   10H        ;SP
                       ;
                       ;BLANK THE DISPLAY
                       ;
0297 119102 BLNK:      LXI  D,BLNKM    ;ADRS OF BLANK MESSAGE TABLE
029A DF                RST  3          ;GET MESSAGE
029B CDE901            CALL DCI        ;SEND TO DISPLAY
029E C9                RET
                       ;
                       ;CONTROL KEY JUMP TABLE
                       ;
029F FEF7   CHDSS:     CPI  0F7H       ;HARDWARE SINGLE STEP PUSHED?
02A1 CAF703            JZ   HDSS       ;HARDWARE SINGLE STEP ROUTINE
02A4 FE86   CINSS:     CPI  86H        ;SOFTWARE SINGLE STEP PUSHED?
02A6 CAF103            JZ   INSS       ;SOFTWARE SINGLE STEP ROUTINE
02A9 FE84   CRUN :     CPI  84H        ;RUN PUSHED?
02AB CAC203            JZ   RUN        ;RUN ROUTINE
02AE FE83   CSTRM:     CPI  83H        ;STORE MEMORY PUSHED?
02B0 CAAF03            JZ   STRM       ;STORE MEMORY ROUTINE
02B3 FE81   CDCRM:     CPI  81H        ;DECREMENT MEMORY PUSHED?
02B5 CAAA03            JZ   DCRM       ;DECREMENT MEMORY ROUTINE
02B8 FE82   CFETA:     CPI  82H        ;FETCH ADRS PUSHED?
02BA CA2B02            JZ   FETA       ;FETCH ADRS ROUTINE
02BD FE85   CFETP:     CPI  85H        ;FETCH PROGRAM COUNTER PUSHED?
02BF CA4101            JZ   TRF7       ;FETCH PROGRAM COUNTER ROUTINE
02C2 FE80   CFETR:     CPI  80H        ;FETCH REGISTER PUSHED?
02C4 CAD502            JZ   FETR       ;FETCH REGISTER ROUTINE
02C7 C9                RET             ;IF NO LEGAL KEY WAS PUSHED
02C8 FE83   CSTRR:     CPI  83H        ;STORE REGISTER PUSHED?
02CA CA1304            JZ   STRR       ;STORE REGISTER ROUTINE
02CD FE81   CDCRR:     CPI  81H        ;DECREMENT REGISTER PUSHED?
02CF CAFD03            JZ   DCRR       ;DECREMENT REGISTER ROUTINE
02D2 C3B802            JMP  CFETA      ;
                       ;
                       ;FETCH REGISTER MODE
```

```
Hex
Adrs Contents  Label   Instruction          Comments

                       ;
02D5 E1        FETR:   POP  H          ;UNDO STACK CALL
02D6 214D02            LXI  H,MA       ;A REG MESSAGE ADRS
02D9 01E70B            LXI  B,SAVA     ;ADRS OF USER A REG CONTENTS
02DC 2B        FETR1:  DCX  H          ;6-2=4 CHARACTERS IN  REG MESSAGE
02DD 2B                DCX  H          ;
02DE 22F80B    FETR2:  SHLD RMF        ;STORE IN REGISTER MESSAGE POINTER
02E1 EB                XCHG            ;PUT MESSAGE ADRS IN DE
02E2 DF                RST  3          ;STORE MESSAGE IN RAM
02E3 0A                LDAX B          ;USERS REG CONTENTS
02E4 5F                MOV  E,A        ;TRANSFER IT TO E
02E5 21F100            LXI  H,LDSP1    ;ADRS WHERE DISPLAY REG DATA GOES
02E8 CD9C03            CALL FETA7      ;FORMAT REG BYTE DATA + STORE IT
02EB 3EDB              MVI  A,0DBH     ;LS BYTE OF SAVIN ADRS
02ED B9                CMP  C          ;IS REG EXAMINED THE INTRP1 ADRS?
02EE CA1003            JZ   FETR6      ;IF IT IS- HEX KEY INPUT NOT LEGAL
02F1 CD4B01    FETR3:  CALL KIND       ;INPUT KEYS
02F4 CDC802            CALL CSTRR      ;LOOK FOR CONTROL
02F7 D2F102            JNC  FETR3      ;IF NOT A HEX KEY OR NEW CONTROL
02FA 5F                MOV  E,A        ;SAVE 1ST HEX KEY IN E
02FB 2C                INR  L          ;POINT TO UDSP1
02FC 3600              MVI  M,0        ;CLEAR IT TO DISPLAY A 0
02FE 2D        FETR4:  DCR  L          ;POINT BACK TO UDSP0
02FF 73                MOV  M,E        ;STORE NEW HEX KEY THERE
0300 CD1903    FETR5:  CALL DPS        ;TO SET DP AND INPUT KEYS
0303 CDC802            CALL CSTRR      ;LOOK FOR CONTROL
0306 D20003            JNC  FETR5      ;IF NOT A HEX KEY OR NEW CONTROL
0309 2C                INR  L          ;POINT TO UDSP1
030A 53                MOV  D,E        ;PUT OLD HEX CHARACTER INTO D
030B 5F                MOV  E,A        ;PUT NEW HEX CHARACTER INTO E
030C 72                MOV  M,D        ;STORE OLD HEX CHARACTER INTO UDSP1
030D C3FE02            JMP  FETR4      ;CONTINUE
0310 CD4B01    FETR6:  CALL KIND       ;INPUT KEYS AND UPDATE DISPLAY
0313 CDC802            CALL CSTRR      ;LOOK FOR CONTROL ONLY
0316 C31003            JMP  FETR6      ;KEEP LOOKING
                       ;
                       ;DECIMAL POINT SET
                       ;
0319 3E01      DPS:    MVI  A,1        ;FLAG SET DATA
031B 32F60B            STA  UDSP6      ;SET DATA MODIFY FLAG
031E CD4B01            CALL KIND       ;GET ANOTHER KEY
0321 F5                PUSH PSW        ;SAVE KEY CODE
0322 AF                XRA  A          ;CLEAR A
0323 32F60B            STA  UDSP6      ;CLEAR DATA MODIFY FLAG
0326 F1                POP  PSW        ;RECOVER KEY CODE
0327 C9                RET
                       ;
                       ;FETCH MEMORY ADDRESS
                       ;
0328 D1        FETA1   POP  D          ;UNDO STACK CALL
0329 114702    FETAR:  LXI  D,FETCH    ;DISPLAY MESSAGE ADRS
032C DF                RST  3          ;STORE MESSAGE IN RAM
032D 0E04              MVI  C,4        ;ADRS DIGIT COUNTER
032F CD4B01    FETA1:  CALL KIND       ;READ KEYS AND SCAN DISPLAY
0332 CDB802            CALL CFETA      ;LOOK FOR CONTROL
0335 D22F03            JNC  FETA1      ;IF NOT A HEX KEY OR NEW CONTROL
0338 21F60B            LXI  H,UDSP6    ;ADRS OF DISPLAY DIGIT 4+2
033B 47                MOV  B,A        ;SAVE HEX KEY INPUT
033C FE01              CPI  1          ;1 KEY PUSHED?
033E C24A03            JNZ  NCTL       ;IF NOT
0341 3E04              MVI  A,4        ;MS ADRS-POSITION VALUE
0343 B9                CMP  C          ;ADRS POSITION POINTER
0344 C24A03            JNZ  NCTL       ;IF 1 KEY NOT MS ADRS BYTE
0347 C32903            JMP  FETAR      ;WAIT FOR ANOTHER KEY
034A 78        NCTL:   MOV  A,B        ;RESTORE NON-1 KEY VALUE
034B 0604              MVI  B,4        ;DISPLAY POSITION COUNTER
034D 2D        FETA2:  DCR  L          ;POINT TO DISP DIGIT ON RIGHT
034E 2D                DCR  L          ;POINT TO DISP DIGIT ON RIGHT
034F 56                MOV  D,M        ;PUT THIS CHARACTER IN D
0350 2C                INR  L          ;POINT TO DISP DIGIT ON LEFT
0351 72                MOV  M,D        ;STORE THE DIGIT SHIFTED 1 TO LEFT
0352 05                DCR  B          ;POSITION COUNTER
0353 C24D03            JNZ  FETA2      ;IF NOT DONE ENTERING ADRS
0356 77                MOV  M,A        ;KEY CODE TO DISP DIGIT 2 ADRS
0357 0D                DCR  C          ;DIGIT COUNTER
0358 C22F03            JNZ  FETA1      ;IF NOT DONE
035B CD9303            CALL FETA6      ;MERGE LS ADRS BYTE IN DISP TO A
035E 4F                MOV  C,A        ;STORE MERGED BYTE IN C
035F 2C                INR  L          ;POINT TO MS ADRS BYTE IN DISP
0360 CD9303            CALL FETA6      ;MERGE IT IN A
0363 47                MOV  B,A        ;STORE IT IN C
0364 21F50B    FETA3:  LXI  H,UDSP5    ;ADRS OF DISP DIGIT 3
0367 5B                MOV  E,E        ;PUT MS ADRS BYTE IN E
0368 CD9C03            CALL FETA7      ;SHPARATE AND STORE IT IN RAM
036B 2D                DCR  L          ;POINT TO UDSP3
036C 59                MOV  E,C        ;SAVE LS ADRS BYTE
036D CD9C03            CALL FETA7      ;SEPARATE AND STORE IT IN RAM
0370 2D                DCR  L          ;POINT TO UDSP1
0371 0A                LDAX B          ;GET DATA AT FETCH ADRS
0372 5F                MOV  E,A        ;PUT IT IN E
0373 CD9C03            CALL FETA7      ;SEPARATE IT AND DISP IT AS DATA
                       ;
                       ;MODIFY THE DATA
                       ;
0376 CD4B01            CALL KIND       ;GET A KEY
0379 CD9F02            CALL CHDSS      ;LOOK FOR ANY CONTROL KEY
037C 5F                MOV  E,A        ;STORE HEX KEY IN E
037D 72                MOV  M,A        ;AND UDSP0
037E 2C                INR  L          ;POINT TO UDSP1
037F 3600              MVI  M,0        ;CLEAR IT
0381 2D        FETA4:  DCR  L          ;POINT TO UDSP0
0382 73                MOV  M,E        ;STORE LATEST KEY ENTRY THERE
0383 CD1903    FETA5:  CALL DPS        ;TO SET DP AND INPUT KEYS
0386 CDAE02            CALL CSTRM      ;LOOK FOR CONTROL KEY
0389 D28303            JNC  FETA5      ;IF NOT A VALID KEY
038C 2C                INR  L          ;POINT TO UDSP1
038D 53                MOV  D,E        ;LAST KEY ENTRY
038E 5F                MOV  E,A        ;NEW  KEY ENTRY
038F 72                MOV  M,D        ;LAST ENTRY SHIFTED TO UDSP1
0390 C38103            JMP  FETA4      ;UPDATE NEW KEY AND CONTINUE
```

Tabella F-2. Listing della ROM (continuazione)

```
Hex
Adrs Contents  Label   Instruction          Comments

                       ;
                       ;MERGES 2 HEX NUMBERS INTO A REG
                       ;
0393 5E        FETA6:  MOV  E,M             ;LS HEX CHAR
0394 23                INX  H               ;POINT TO MS HEX CHAR
0395 7E                MOV  A,M             ;MS HEX CHAR
0396 07                RLC                  ;MOVE IT TO 4 MS BITS IN A
0397 07                RLC                  ;AND CLEAR  4 LS BITS IN A
0398 07                RLC                  ;
0399 07                RLC                  ;
039A B3                ORA  E               ;MERGE MS AND LS HEX CHARS IN A
039B C9                RET
                       ;
                       ;SEPARATES 2 HEX CHARACTERS IN A + STORES IN M
                       ;
039C 7B        FETA7:  MOV  A,E             ;PUT MERGED HEX CHARS INTO A
039D 0F                RRC                  ;MOVE MS HEX CHAR TO 4 LS BITS A
039E 0F                RRC                  ;
039F 0F                RRC                  ;
03A0 0F                RRC                  ;
03A1 160F              MVI  D,0FH           ;MASK TO CLEAR 4 MS BITS OF A
03A3 A2                ANA  D               ;DO IT
03A4 77                MOV  M,A             ;STORE MS HEX CHAR
03A5 2D                DCR  L               ;LS CHAR DESTINATION ADRS
03A6 7B                MOV  A,E             ;GET MERGED HEX CHARS
03A7 A2                ANA  D               ;CLEAR MS HEX CHAR
03A8 77                MOV  M,A             ;STORE LS HEX CHAR
03A9 C9                RET
                       ;
                       ;DECREMENTS MEMORY ADRS
                       ;
03AA 0B        DCRM:   DCX  B               ;POINT TO NEXT LOWER ADRS
03AB E1                POP  H               ;UNDO STACK CALL
03AC C36403            JMP  FETA3           ;TO FETCH NEW ADRS DATA
                       ;
                       ;STORES DATA IN MEM AND INCREMENTS ADRS
                       ;
03AF D1        STRM:   POP  D               ;UNDO STACK CALL
03B0 CD9303            CALL FETA6           ;MERGE 2 HEX DATA VALUES INTO A
03B3 5F                MOV  E,A             ;SAVE IN E
03B4 02                STAX B               ;STORE DATA BYTE IN RAM ADRS IN B
03B5 0A                LDAX B               ;READ IT BACK
03B6 BB                CMP  E               ;IS IT THE SAME? DID IT STORE?
03B7 03                INX  B               ;POINT TO NEXT RAM ADRS
03B8 CA6403            JZ   FETA3           ;FETCH NEW ADRS IF LAST STORE OK
03BB 0B                DCX  B               ;ELSE POINT BACK TO LAST RAM ADRS
03BC C5                PUSH B               ;SAVE THIS ADRS
03BD D7                RST  2               ;BEEP A FAIL TO STORE ERROR
03BE C1                POP  B               ;RESTORE RAM ADRS
03BF C36403            JMP  FETA3           ;AND FETCH IT AGAIN
                       ;
                       ;RUNS THE PROGRAM STARTING AT ADRS IN DISPLAY
                       ;
03C2 213201 RUN:       LXI  H,0132H         ;INSTR CODES FOR STA 01XX (RUN)
03C5 22D50B RUN1:      SHLD RAML1           ;JUMP LINK IN RAM
03C8 2110C3            LXI  H,0C310H        ;INSTR CODES FOR 00 AND JMP XXXX
03CB 22D70B            SHLD RAML2           ;JUMP LINK IN RAM
03CE 31DB0B            LXI  SP,SAVIM        ;USER PROG START ADRS LINK+1
03D1 C5                PUSH B               ;STORE USER START ADRS IN RAM LINK
03D2 21DF0B            LXI  H,SAVSH         ;RAM ADRS OF MSBYTE
03D5 360B              MVI  M,0BH           ;MSBYTE USER SP
03D7 2B                DCX  H               ;RAM ADRS OF LSBYTE USER SP
03D8 7E                MOV  A,M             ;LSBYTE USER SP
03D9 FE40              CPI  40H             ;<= 40
03DB D2E003            JNC  RUN2            ;IF AVAIL STACK SPACE
03DE 36B0              MVI  M,0B0H          ;IF NOT, RESET POINTER
03E0 31E20B RUN2:      LXI  SP,SAVE         ;PREPARE TO RESTORE USER REGS
03E3 D1                POP  D               ;RESTORE D,E
03E4 C1                POP  B               ;RESTORE B,L
03E5 F1                POP  PSW             ;RESTORE PSW,A
03E6 31DE0B            LXI  SP,SAVSL        ;RAM ADRS OF USER SP
03E9 E1                POP  H               ;PUT SPH,L IN H,L
03EA F9                SPHL                 ;TRANSFER SP VAL TO CPU SP
03EB 2AE00B            LHLD SAVL            ;RESTORE H,L
03EE C3D50B            JMP  RAML1           ;LINK TO USER PROGRAM
                       ;
                       ;INSTRUCTION SINGLE STEP AND RETURN TO MONITOR
                       ;
03F1 213206 INSS:      LXI  H,0632H         ;INSTR CODES FOR STA 06XX (INSS)
03F4 C3C503            JMP  RUN1            ;SET LINKS,RESTORE REGS,USER PROG
                       ;
                       ;HARDWARE SINGLE STEP ONE LINE OF CODE
                       ;
03F7 213203 HDSS:      LXI  H,0332H         ;INSTR CODES FOR STA 03XX (HDSS)
03FA C3C503            JMP  RUN1            ;SET LINKS,RESTORE REGS,USER PROG
                       ;
                       ;DECREMENTS REGISTER DISPLAYED
                       ;
03FD D1        DCRR:   POP  D               ;UNDO STACK CALL
03FE 03                INX  B               ;POINT TO LAST SAVX REG RAM ADRS
03FF 2AF80B            LHLD RMF             ;REGISTER MESSAGE POINTER
0402 2B                DCX  H               ;SUBTRACT 2 FROM IT
0403 2B                DCX  H               ;
0404 3EE8              MVI  A,0E8H          ;LS BYTE OF SAVA+1 ADRS IN RAM
0406 B9                CMP  C               ;SEE IF IT'S IM REG
0407 C2DC02            JNZ  FETR1           ;FETCH NEW NON-IM REG+DISP IF NOT
040A 01DB0B            LXI  B,SAVIM         ;ADRS OF IM VALUE IN RAM
040D 217D02            LXI  H,IM            ;ADRS OF IM DISP MESSAGE
0410 C3DC02            JMP  FETR1           ;FETCH IM REG + DISP IF IT IS IM
                       ;
                       ;STORES REGISTER DATA AND INCREMENTS
                       ;
0413 D1        STRR:   POP  D               ;UNDO STACK CALL
0414 CD9303            CALL FETA6           ;MERGE 2 HEX DATA VALUES IN A
0417 02                STAX B               ;STORE DATA BYTE IN SAVX REG ADRS
0418 0B                DCX  B               ;POINT TO NEXT SAVX REG ADRS
0419 2AF80B            LHLD RMF             ;REGISTER MESSAGE POINTER
041C 23                INX  H               ;POINT TO NEXT REG MESSAGE
041D 23                INX  H               ;
```

```
Hex
Adrs Contents  Label   Instruction          Comments

041E 23                INX  H               ;
041F 23                INX  H               ;
0420 3EDA              MVI  A,0DAH          ;LS BYTE OF SAVIM-1 ADRS IN RAM
0422 B9                CMP  C               ;SEE IF IT'S A REG
0423 CAD502            JZ   FETR            ;FETCH A REG + DISP IF IT IS
0426 C3DE02            JMP  FETR2           ;FETCH NEXT REG + DISP IF NOT
                       ;
                       ;DELAYS APPROX 1MS
                       ;
0429 C5        DELA:   PUSH B
042A 010100            LXI  B,0001H         ;FIXED 1MS VALUE
042D C33104            JMP  DEL1            ;START TIMING LOOP
                       ;
                       ;DELAYS APPROX 1MS TIMES VALUE IN BC
                       ;
0430 C5        DELB:   PUSH B
0431 F5        DEL1:   PUSH PSW
0432 AF                XRA  A               ;CLEAR A
0433 D5                PUSH D               ;
0434 168C      DEL2:   MVI  D,TIME          ;1MS SMALL LOOP TIME CONSTANT
0436 15        DEL3:   DCR  D               ;1MS LOOP
0437 C23604            JNZ  DEL3            ;IF NOT 1MS WORTH OF COUNTS
043A 0B                DCX  B               ;LARGE LOOP COUNTER
043B B8                CMP  B               ;MS BYTE =0?
043C C23404            JNZ  DEL2            ;IF NOT, LOOP FOR 1 MORE MS
043F B9                CMP  C               ;LS BYTE =0?
0440 C23404            JNZ  DEL2            ;IF NOT, LOOP
0443 D1                POP  D               ;WHEN DONE
0444 F1                POP  PSW
0445 C1                POP  B
0446 C9                RET
                       ;
                       ;BEEP PRODUCES A FIXED TONE FREQ. + DURATION
                       ;
0447 2EFF      BEEP2:  MVI  L,0FFH          ;DURATION MULTIPLIER
0449 2600              MVI  H,0             ;SPEAKER FLAG
044B 48                MOV  C,B             ;SET COUNT REG B
044C 5A                MOV  E,D             ;SET COUNT REG E
044D E5                PUSH H               ;INIT. DONE FLAG
044E 0D        BEEP3:  DCR  C               ;DECR FREQ
044F C28004            JNZ  BEEP8           ;
0452 48                MOV  C,B             ;RESTORE FREQ
0453 7C                MOV  A,H             ;CHG SPKR FLAG
0454 2F                CMA                  ;
0455 B7                ORA  A               ;
0456 67                MOV  H,A             ;
0457 C26004            JNZ  BEEP4           ;TEST SPKR FLAG
045A 3EC0              MVI  A,0C0H          ;TURN SPKR OFF
045C 30                SIM                  ;
045D C36404            JMP  BEEP5           ;
0460 3E40      BEEP4:  MVI  A,40H           ;TURN SPKR ON
0462 30                SIM                  ;
0463 BE                CMP  M               ;TIME DELAY INSTR
0464 F1        BEEP5:  POP  PSW             ;GET DONE FLAG
0465 F5                PUSH PSW             ;
0466 B7                ORA  A               ;
0467 CA6F04            JZ   BEEP6           ;CONTINUE IF NOT DONE
046A 7C                MOV  A,H             ;RETURN IF SPKR OFF
046B B7                ORA  A               ;
046C CA7E04            JZ   BEEP7           ;
046F 1D        BEEP6:  DCR  E               ;DURATION CNTR
0470 C28804            JNZ  BEEP9           ;
0473 5A                MOV  E,D             ;RESTORE DURATION
0474 2D                DCR  L               ;TIME COUNT
0475 C24E04            JNZ  BEEP3           ;
0478 F1                POP  PSW             ;GET DONE FLAG
0479 2F                CMA                  ;
047A F5                PUSH PSW             ;SET DONE FLAG
047B C34E04            JMP  BEEP3           ;
047E F1        BEEP7:  POP  PSW             ;
047F C9                RET                  ;
0480 E3        BEEP8:  XTHL                 ;DELAY 81 CYCLES
0481 E3                XTHL                 ;
0482 E3                XTHL                 ;
0483 E3                XTHL                 ;
0484 BE                CMP  M               ;
0485 C36F04            JMP  BEEP6           ;
0488 00        BEEP9:  NOP                  ;DELAY 14 CYCLES
0489 00                NOP                  ;
048A C34E04            JMP  BEEP3           ;
                       ;
                       ;SIGNATURE ANALYSIS TEST LOOP
                       ;
048D F3        SATL1:  DI                   ;TURN OFF INTERRUPTS
048E DB80              IN   80H             ;PULSE A15 READ START-STOP LINE
0490 D380              OUT  80H             ;PULSE A15 WRITE START-STOP LINE
0492 31CE0B            LXI  SP,0BCEH        ;SET TO MONITOR VALUE
                       ;
                       ;RAM PROTECT TEST
                       ;
0495 D311              OUT  11H             ;SET RAM PROTECT
0497 32110B            STA  0B11H           ;WRITE TO UNPROTECTED RAM
049A 321109            STA  0911H           ;WRITE TO PROTECTED RAM
049D D310              OUT  10H             ;UNPROTECT RAM
                       ;
                       ;OUTPUT PORT TEST
                       ;
049F AF                XRA  A               ;CLEAR A
04A0 37                STC                  ;SET CARRY BIT TO 1 FOR 8 LOOPS
04A1 17        SATL2:  RAL                  ;MOVE 1 BIT TO LEFT
04A2 D330              OUT  LOUT            ;OUTPUT PORT LEDS
04A4 D2A104            JNC  SATL2           ;IF NOT DONE
                       ;
                       ;DISPLAY LATCH TEST
                       ;
04A7 17        SATL3:  RAL                  ;MOVE 1 BIT TO LEFT
04A9 D338              OUT  DSP             ;OUTPUT DISPLAY SEGMENTS
04AB D2A704            JNC  SATL3           ;IF NOT DONE
                       ;
```

Tabella F-2. Listing della ROM (continuazione)

```
Hex
Adrs Contents Label   Instruction          Comments

                      ;SCAN LATCH TEST
                      ;
04AD 2F               CMA                  ;SET A TO FF
04AE 3F               CMC                  ;CLEAR CARRY
04AF 17      SATL4:   RAL                  ;MOVE 0 BIT TO LEFT
04B0 D328             OUT  SCAN            ;OUTPUT SCAN LINES
04B2 CD2904           CALL DELB            ;STRETCH EACH DISP DIGIT
04B5 DAAF04           JC   SATL4           ;IF NOT DONE
                      ;
                      ;SPEAKER SERIAL OUT TEST
                      ;
04B8 3E40             MVI  A,40H           ;SPEAKER ON MASK
04BA 30               SIM                  ;TURN SPKR ON
04BB 3EC0             MVI  A,0C0H          ;SPKR OFF MASK
04BD 30               SIM                  ;TURN SPKR OFF
                      ;
                      ;KEY INPUT TEST
                      ;
04BE AF               XRA  A               ;CLEAR A
04BF D328             OUT  SCAN            ;ALL SCAN LINES TO LOGIC 0
04C1 DB18             IN   KEY             ;RESPOND TO ALL KEYS
                      ;
                      ;INPUT PORT TEST
                      ;
04C3 DB20             IN   SIN             ;INPUT THE INPUT SWITCHES
                      ;
                      ;INTERRUPT KEY TEST
                      ;
04C5 211000           LXI  H,BEEP          ;ADRS OF BEEP ROUTINE
04C8 22FD0A           SHLD 0AFDH           ;STORE IT IN INTRPT RAM LINK 6.5
04CB 3EC3             MVI  A,0C3H          ;JUMP OP CODE
04CD 32FC0A           STA  0AFCH           ;STORE IT IN INTRPT RAM LINK 6.5
04D0 3E1D             MVI  A,1DH           ;INTRPT MASK FOR RST 6.5
04D2 30               SIM                  ;SET MASK
04D3 FB               EI                   ;ENABLE INTERRUPTS
04D4 C38D04           JMP  SATLI           ;LOOP BACK TO START OVER AGAIN
                      ;
                      ;PRESENTS INPUT SWITCH DATA ON OUTPUT LEDS
                      ;
04D7 3A0020 ECHO:     LDA  2000H           ;READ INPUT PORT SWITCHES
04DA 320030           STA  3000H           ;WRITE TO OUTPUT PORT LEDS
04DD C3D704           JMP  ECHO            ;REPEAT
                      ;
                      ;AND GATE PROGRAM
                      ;
04E0 3A0020 ANDGT:    LDA  2000H           ;READ INPUT PORT
04E3 FEFF             CPI  0FFH
04E5 CAF004           JZ   ON              ;JUMP IF ALL*BITS ARE ONE
04E8 3EFF             MVI  A,0FFH
04EA 320030           STA  3000H           ;TURN OUTPUT LEDS OFF
04ED C3E004           JMP  ANDGT
04F0 3EFE    ON:      MVI  A,0FEH          ;TURN LED ON
04F2 320030           STA  3000H
04F5 C3E004           JMP  ANDGT
                      ;
                      ;CONVEYOR BELT CONTROLLER DEMO PROGRAM
                      ;
04F8 CD9702 CONV:     CALL BLNK            ;BLANK THE DISPLAY
04FB AF               XRA  A               ;CLEAR A
04FC 32F008           STA  UDSP0           ;SET DISPLAY COUNTER
04FF CD4B01 LOOP:     CALL KIND            ;DISPLAY MESSAGE & READ KEYS
0502 FE00             CPI  0               ;CHECK FOR "0" KEY
0504 C2FF04           JNZ  LOOP
0507 21F00B           LXI  H,NUM           ;INCREMENT COUNT
050A 34               INR  M
050B 7E               MOV  A,M             ;TEST FOR COUNT=10
050C FE0A             CPI  10
050E CA1405           JZ   MOVE
0511 C3FF04           JMP  LOOP
0514 3E7F    MOVE:    MVI  A,7FH
0516 16F0             MVI  D,MIN           ;SET LOW FREQ VALUE
0518 320030 MLP:      STA  3000H           ;WRITE DATA TO LEDS
051B 015000           LXI  B,DTIME         ;WAIT DELAY TIME
051E D5               PUSH D
051F CD3004           CALL DELB
0522 D1               POP  D
0523 CD3105           CALL TONE            ;GENERATE BEEP
0526 37               STC
0527 1F               RAR                  ;SHIFT PATTERN
0528 DA1805           JC   MLP
052B 320030           STA  3000H           ;TURN OFF LEDS
052E C3F004           JMP  CONV
0531 5F      TONE:    MOV  E,A             ;SAVE A
0532 D5               PUSH D
0533 42               MOV  B,D
0534 CD1200           CALL BEEP1           ;GENERATE BEEP
0537 D1               POP  D
0538 7A               MOV  A,D             ;INCREASE FREQUENCY
0539 D610             SUI  INCR
053B 57               MOV  D,A
053C 7B               MOV  A,E             ;RESTORE A
053D C9               RET
00F0         NUM      EQU  0BF0H           ;RAM BUFFER LOCATION
0010         INCR     EQU  10H             ;FREQ INCREMENT
00F0         MIN      EQU  0F0H            ;FREQUENCY MINIMUM
0050         DTIME    EQU  50              ;TIME BETWEEN SHIFTS
                      ;
                      ;WELL TEMPERED MICROPROCESSOR
                      ;GENERATES PSEUDO RANDOM TONES FROM ROM
                      ;
053E 218001 WTM:      LXI  H,0180H         ;SET ROM POINTER TO 0180
0541 7E      WTM1:    MOV  A,M             ;ROM CONTENTS
0542 E67E             ANI  7EH             ;MASK BITS
0544 47               MOV  B,A             ;SET BEEP FREQ. REG
0545 E5               PUSH H               ;SAVE ADRS POINTER
0546 CD1200           CALL BEEP1           ;GENERATE BEEP
0549 E1               POP  H               ;RESTORE ADRS POINTER
054A 015000           LXI  B,0050H         ;SET DELAY REG
```

```
Hex
Adrs Contents Label   Instruction          Comments

054D CD3004           CALL DELB            ;PAUSE
0550 23               INX  H               ;NEXT ADRS OF ROM
0551 3E03             MVI  A,03H           ;LAST ADRS OF LOOP
0553 BC               CMP  H               ;LAST ADRS
0554 CA3E05           JZ   WTM             ;IF LAST ADRS
0557 C34105           JMP  WTM1            ;IF NOT LAST ADRS
                      ;
                      ;SQUIRREL  FEEDBACK SHIFT REGISTER DISPLAY
                      ;
055A CD9702 SQRL:     CALL BLNK            ;CLEAR THE DISPLAY
055D AF               XRA  A               ;CLEAR A AND CARRY
055E 0601             MVI  B,1             ;SEED
0560 17      SQRL1:   RAL                  ;SHIFT A  A7 TO CARRY
0561 57               MOV  D,A             ;SAVE IT
0562 78               MOV  A,B             ;GET B FSR REGISTER
0563 1F               RAR                  ;SHIFT A7 INTO B7
0564 4F               MOV  C,A             ;SAVE IT
0565 A8               XRA  B               ;XOR B0 AND B1
0566 E601             ANI  1               ;SET B7 TO B1 TO 0
0568 B2               ORA  D               ;INSERT B0 XOR B1 IN A0
0569 41               MOV  B,C             ;RESTORE NEW B
056A 21FC0B           LXI  H,IDSP2         ;ADRS OF DECODED DISPLAY D2
056D F5               PUSH PSW             ;SAVE A FSR REG
056E 0F               RRC                  ;GET DIGIT 2 BITS IN POSITION
056F 0F               RRC                  ;
0570 CD8605           CALL SQRL3           ;FORMAT AND STORE DIGITS 2-3
0573 2C               INR  L               ;ADRS DIGIT 4
0574 78               MOV  A,B             ;GET B FSR REGISTER
0575 07               RLC                  ;GET DIGIT 4 BITS IN POSITION
0576 CD8605           CALL SQRL3           ;FORMAT AND STORE DIGITS 4-5
0579 0E0F             MVI  C,0FH           ;DISPLAY TIMER SEGMENTS
057B CDC801 SQRL2:    CALL SDS             ;DISPLAY SEGMENTS
057E 0D               DCR  C               ;DCR TIMER
057F C27B05           JNZ  SQRL2           ;IF TIME NOT UP
0582 F1               POP  PSW             ;RESTORE A FSR REGISTER
0583 C36005           JMP  SQRL1           ;DO IT AGAIN
0586 4F      SQRL3:   MOV  C,A             ;SAVE SHIFTED VALUE
0587 F6F8             ORI  0F8H            ;BLANK E,F,G,DP SEGS
0589 77               MOV  M,A             ;STORE IN DIG 2 OR 4
058A 79               MOV  A,C             ;RECALL SHIFTED VALUE
058B 2C               INR  L               ;ADRS OF DIG 3-5
058C 07               RLC                  ;GET A SEG IN D0
058D E601             ANI  01              ;CLEAR D7-D1
058F 57               MOV  D,A             ;SAVE A SEG
0590 79               MOV  A,C             ;RECALL SHIFTED VALUE
0591 0F               RRC                  ;MOVE A,F,E,D TO D6-D3
0592 E638             ANI  38H             ;CLEAR D7-D6,D2-D0
0594 B2               ORA  D               ;INSERT A SEG IN D0
0595 F6C6             ORI  0C6H            ;BLANK B,C,H,DP SEGS
0597 77               MOV  M,A             ;STORE IN DIG 3 OR 5
0598 C9               RET
                      ;
                      ;ORGAN  GENERATES TONES FROM KEYBOARD
                      ;
0599 1640    ORGAN:   MVI  D,40H           ;INITIALIZE SPKR FLAG
059B CD9A01 ORGAN1:   CALL KRI             ;READ KEYS
059E CDBB05           CALL CODE            ;DECODE KEYS & LOOK-UP DELAY VALUE
05A1 B7               ORA  A               ;CHECK FOR NO KEY
05A2 CA9B05           JZ   ORGAN1
05A5 CDAE05           CALL DLY             ;TIME DELAY
05A8 CDB505           CALL SPKR            ;CHANGE SPEAKER STATE
05AB C39B05           JMP  ORGAN1          ;REPEAT
                      ;
                      ;DELAY ROUTINE
                      ;
05AE 3D      DLY:     DCR  A               ;DECREMENT A UNTIL ZERO
05AF 00               NOP
05B0 00               NOP
05B1 C2AE05           JNZ  DLY
05B4 C9               RET
                      ;
                      ;SPKR ROUTINE TO CHANGE SPEAKER STATE
                      ;
05B5 7A      SPKR:    MOV  A,D             ;GET SPKR FLAG
05B6 EE80             XRI  80H             ;CONPLEMENT BIT 7
05B8 57               MOV  D,A             ;SAVE FLAG
05B9 30               SIM                  ;OUTPUT TO SPKR
05BA C9               RET
                      ;
                      ;CODE ROUTINE DETERMINES WHICH KEY IS
                      ; PRESSED AND LOOKS UP DELAY VALUE
                      ;
05BB 0607    CODE:    MVI  B,7             ;INITIALIZE ROW COUNT
05BD 21EF0B           LXI  H,0BEFH         ;INITIALIZE DATA ADDRESS
05C0 7E      READ:    MOV  A,M             ;GET KEY DATA
05C1 2F               CMA
05C2 B7               ORA  A               ;ANY KEY PRESSED?
05C3 CAD805           JZ   NOKEY
05C6 FE04             CPI  4H              ;DATA=100?
05C8 C2CC05           JNZ  SHIFT           ;IF YES CHANGE TO 011
05CB 3D               DCR  A
05CC 4F      SHIFT:   MOV  C,A             ;SAVE KEY DATA
05CD 78               MOV  A,B             ;SHIFT ROW COUNT
05CE 07               RLC
05CF 07               RLC
05D0 B1               ORA  C               ;COMBINE ROW COUNT & DATA
05D1 21D905           LXI  H,TABLE-9       ;SET LOOK-UP ADDRESS
05D4 85               ADD  L
05D5 6F               MOV  L,A
05D6 7E               MOV  A,M             ;GET DELAY VALUE
05D7 C9               RET
05D8 05      NOKEY:   DCR  B               ;NO KEY PRESSED-GO TO NEXT ROW
05D9 2B               DCX  H
05DA 78               MOV  A,B             ;DONE?
05DB FE01             CPI  1
05DD C2C005           JNZ  READ            ;IF NOT, READ NEXT ROW
05E0 AF               XRA  A               ;NO KEY - SET DELAY TO 0
05E1 C9               RET
                      ;
```

Tabella F-2. Listing della ROM (continuazione)

```
Hex
Adrs Contents Label     Instruction          Comments

                        ;LOOK-UP TABLE FOR ORGAN DELAY VALUES
                        ;
05E2 E6       TABLE:    DB   0E6H      ;E3
05E3 00                 DB   0
05E4 00                 DB   0
05E5 00                 DB   0
05E6 DA                 DB   0DAH     ;F3
05E7 BD                 DB   0BDH     ;G3
05E8 A3                 DB   0A3H     ;A3
05E9 00                 DB   0
05EA 8F                 DB   8FH      ;B3
05EB 85                 DB   85H      ;C4
05EC 72                 DB   72H      ;D4
05ED 00                 DB   0
05EE 64                 DB   64H      ;E4
05EF 5C                 DB   5CH      ;F4
05F0 4D                 DB   4DH      ;G4
05F1 00                 DB   0
05F2 43                 DB   43H      ;A4
05F3 38                 DB   38H      ;B4
05F4 33                 DB   33H      ;C5
05F5 00                 DB   0
05F6 2D                 DB   2DH      ;D5
05F7 24                 DB   24H      ;E5
05F8 20                 DB   20H      ;F5
                        ;
                        ;ROCKET BLAST-OFF DEMO PROGRAM
                        ;
05F9 3EAA     ROCT:     MVI  A,0AAH   ;ALL AMBER LED MASK
05FB D330               OUT  LOUT     ;TURN ON AMBER LEDS
05FD 115C06             LXI  D,ROCM   ;ROCKET DISPLAY MESSAGE
0600 DF                 RST  3        ;STORE MESSAGE IN RAM
0601 0680     ROCT1:    MVI  B,50H    ;1 SECOND DELAY LOOP COUNTER
0603 CDE901   ROCT2:    CALL DCI      ;UPDATE DISPLAY
0606 05                 DCR  B        ;COUNTER
0607 C20306             JNZ  ROCT2    ;REPEAT IF <1 SECOND
060A D7                 RST  2        ;BEEP
060B 21F00B             LXI  H,UDSP0  ;COUNT DOWN SECONDS DIGIT
060E 35                 DCR  M        ;DECREMENT IT
060F C20106             JNZ  ROCT1    ;IF NOT LAST COUNT (0 SECONDS)
0612 0600               MVI  B,0      ;START BLAST-OFF FREQUENCY
0614 1602               MVI  D,2      ;START BLAST-OFF DURATION
0616 3EE7     ROCT3:    MVI  A,0E7H   ;LED PATTERN
0618 CD5506             CALL ROCT5    ;SEQUENCE
061B 3EDB               MVI  A,0DBH   ;LED PATTERN
061D CD5506             CALL ROCT5    ;SEQUENCE
0620 3EBD               MVI  A,0BDH   ;LED PATTERN
0622 CD5506             CALL ROCT5    ;SEQUENCE
0625 3E7E               MVI  A,07EH   ;LED PATTERN
0627 CD5506             CALL ROCT5    ;SEQUENCE
062A C21606             JNZ  ROCT3    ;REPEAT IF B70, FREQ <MAX
062D AF                 XRA  A        ;CLEAR A
062E 2F                 CMA           ;SET A TO FF
062F D330               OUT  LOUT     ;TURN OFF ALL LEDS
0631 010005             LXI  B,0500H  ;1 SECOND DELAY VALUE
0634 CD3004             CALL DELB     ;FOR 1 SECOND PAUSE
0637 3E03               MVI  A,3      ;NOTE COUNTER
0639 0634     ROCT4:    MVI  B,34H    ;1ST 3 NOTE FREQ
063B F5                 PUSH PSW      ;SAVE NOTE COUNT
063C CD1200             CALL BEEP1    ;PLAY NOTE
063F F1                 POP  PSW      ;RESTORE NOTE COUNT
0640 010000             LXI  B,0090H  ;PAUSE VALUE
0643 CD3004             CALL DELB     ;PAUSE BETWEEN NOTES
0646 3D                 DCR  A        ;NOTE COUNTER
0647 C23906             JNZ  ROCT4    ;IF NOT LAST OF 3 FIRST NOTES
064A 0685               MVI  B,85H    ;LAST NOTE FREQ
064C 1640               MVI  D,40H    ;LAST NOTE DURATION
064E CD4704             CALL BEEP2    ;PLAY LAST NOTE
0651 CF                 RST  1        ;RETURN TO MONITOR
0652 C3F905             JMP  ROCT     ;RUN PROGRAM AGAIN
0655 D330     ROCT5:    OUT  LOUT     ;UPDATE LEDS
0657 CD4704             CALL BEEP2    ;BEEP
065A 05                 DCR  B        ;INCREASE FREQ BY 1
065B C9                 RET           ;
065C 09       ROCM:     DB   09H      ;9
065D 1C                 DB   1CH      ;-
065E 18                 DB   18H      ;SMALL C
065F 15                 DB   15H      ;SMALL O
0660 1B                 DB   1BH      ;SMALL R
0661 10                 DB   10H      ;SPACE
                        ;
                        ;STOPWATCH
                        ;
0662 11BC06   STW:      LXI  D,STWM   ;ADRS OF DISPLAY MESSAGE
0665 DF                 RST  3        ;ZERO DISPLAY
0666 11FF00             LXI  D,00FFH  ;RUN AND 3-KEY STATUS
0669 AF       STW1:     XRA  A        ;CLEAR A
066A 32F60B             STA  UDSP6    ;TURN OFF DECIMAL POINT FLAG
066D CDE901             CALL DCI      ;UPDATE DISPLAY
0670 06F5               MVI  B,0F5H   ;DELAY CONSTANT
0672 05       STWL:     DCR  B        ;DELAY COUNTER
0673 29                 DAD  H        ;DELAY INSTRUCTION
0674 C27206             JNZ  STWL     ;IF NOT DONE
0677 3EFB               MVI  A,0FBH   ;0-KEY SCAN DATA
0679 D328               OUT  SCAN     ;0-KEY ROW
067B DB18               IN   KEY      ;INPUT
067D FEFE               CPI  0FEH     ;CHECK FOR 0-KEY
067F CA6206             JZ   STW      ;IF PUSHED
0682 3EF7               MVI  A,0F7H   ;3-KEY SCAN DATA
0684 D328               OUT  SCAN     ;3-KEY ROW
0686 DB18               IN   KEY      ;INPUT
0688 FEFB               CPI  0FBH     ;CHECK FOR 3-KEY
068A C2B706             JNZ  STW6     ;IF NOT PUSHED
068D BB                 CMP  E        ;WAS IT PUSHED BEFORE?
068E CA9306             JZ   STW2     ;IF IT WAS
0691 5F                 MOV  E,A      ;PUSHED STATUS CODE
0692 14                 INR  D        ;CHANGE RUN MODE START/STOP
0693 7A       STW2:     MOV  A,D      ;CURRENT RUN MODE
0694 1F                 RAR           ;FLAG MODE BIT TO CARRY
0695 D26906             JNC  STW1     ;IF IN STOP MODE
0698 21F00B             LXI  H,UDSP0  ;.01 SEC MEM LOCATION
069B 34       STW3:     INR  M        ;INCREMENT DIGIT COUNT
069C 7E                 MOV  A,M      ;LOAD DIGIT IN A
069D FE0A               CPI  0AH      ;OVERFLOW
069F C2A806             JNZ  STW5     ;IF DIGIT 9 OR LESS
06A2 3600               MVI  M,0      ;SET DIGIT TO 0
06A4 2C       STW4:     INR  L        ;NEXT HIGHER DIGIT ADRS
06A5 C39B06             JMP  STW3     ;REPEAT SEQUENCE ON THIS DIGIT
06A8 21F30B   STW5:     LXI  H,UDSP3  ;10 SEC MEM LOCATION
06AB 7E                 MOV  A,M      ;LOAD DIGIT IN A
06AC FE06               CPI  6        ;OVERFLOW
06AE C26906             JNZ  STW1     ;IF DIGIT 5 OR LESS
06B1 3600               MVI  M,0      ;SET DIGIT TO 0
06B3 2C                 INR  L        ;MIN DIGIT ADRS
06B4 C3A406             JMP  STW4     ;REPEAT SEQUENCE FOR MIN DIGIT
06B7 1EFF     STW6:     MVI  E,0FFH   ;3-KEY NOT PUSHED CODE
06B9 C39306             JMP  STW2     ;CONTINUE IN CURRENT RUN MODE
06BC 00       STWM:     DB   0        ;ZERO DISPLAY MESSAGE TABLE
06BD 00                 DB   0
06BE 00                 DB   0
06BF 00                 DB   0
06C0 10                 DB   10H
06C1 00                 DB   0
                        ;
                        ;SNAKE PADDLE GAME
                        ;
06C2 2100FF   SNAKE:    LXI  H,0FF00H ;INITIALIZE SCORE DISPLAY
06C5 06F7               MVI  B,RIGHT  ;RIGHT PLAYER CODE
06C7 E5                 PUSH H        ;SAVE SCORE
06C8 3E40     SERV:     MVI  A,40H    ;SERVE SPEED
06CA 32000B             STA  SPEED    ;CURRENT BALL SPEED
06CD CD9702             CALL BLNK     ;CLEAR THE DISPLAY
06D0 E1                 POP  H        ;GET SCORE
06D1 0E1C               MVI  C,1CH    ;CODE FOR "-" CHAR
06D3 78                 MOV  A,B      ;PLAYER CODE
06D4 FEF7               CPI  RIGHT    ;RT PLAYER LOST POINT?
06D6 79                 MOV  A,C      ;DASH CHAR CODE
06D7 C2E406             JNZ  LLOSE    ;IF LEFT PLAYER LOST POINT
06DA 32F50B             STA  UDSP5    ;LEFT SERVE DISPLAY MESSAGE
06DD 24                 INR  H        ;LEFT PLAYER GETS POINT
06DE 7C                 MOV  A,H      ;LEFT SCORE
06DF 06FB               MVI  B,LEFT   ;LEFT PLAYER GETS SERVE
06E1 C3EB06             JMP  SERV1    ;CONTINUE
06E4 32F20B   LLOSE:    STA  UDSP2    ;RIGHT SERVE DISPLAY MESSAGE
06E7 2C                 INR  L        ;RIGHT PLAYER GETS POINT
06E8 7D                 MOV  A,L      ;RIGHT SCORE
06E9 06F7               MVI  B,RIGHT  ;RIGHT PLAYER GETS SERVE
06EB FE0A     SERV1:    CPI  0AH      ;LAST POINT
06ED E5                 PUSH H        ;SAVE SCORE
06EE 22F00B             SHLD UDSP0    ;DISPLAY SCORE
06F1 CDE901             CALL DCI      ;UPDATE DISPLAY
06F4 C20007             JNZ  SERV2    ;IF NOT LAST GAME POINT
06F7 CD9907   ENDGM:    CALL RSTGM    ;NEW GAME REQUEST?
06FA CDC801             CALL SDS      ;UPDATE DISPLAY
06FD C2F706             JNZ  ENDGM    ;WAIT FOR NEW GAME REQUEST
0700 CD9907   SERV2:    CALL RSTGM    ;NEW GAME REQUEST?
0703 CAC206             JZ   SNAKE    ;IF NEW GAME REQUESTED
0706 CDA207             CALL PADL     ;CHECK PADDLE
0709 C20007             JNZ  SERV2    ;IF NOT PUSHED
070C C5       RBND:     PUSH B        ;SAVE HITTING PLAYER CODE
070D 78                 MOV  A,B      ;HITTING PLAYER CODE
070E FEFB               CPI  LEFT     ;LEFT PLAYER CODE
0710 01D407             LXI  B,LSM    ;LEFT PLAYER MESSAGE POINTER
0713 CA1907             JZ   RSRV     ;IF LEFT PLAYER WAS HITTER
0716 01E207             LXI  B,RSM    ;RIGHT PLAYER MESSAGE CODE
0719 CD3807   RSRV:     CALL PASS     ;START BALL MOVING
071C C1                 POP  B        ;RESTORE LAST HITTER CODE
071D 78                 MOV  A,B      ;HITTING PLAYER CODE
071E FEFB               CPI  LEFT     ;LEFT PLAYER CODE
0720 06F7               MVI  B,RIGHT  ;RIGHT PLAYER CODE
0722 CA2707             JZ   RBND1    ;IF BALL CAME FROM LEFT PLAYER
0725 06FB               MVI  B,LEFT   ;LEFT PLAYER NEXT HITTER
0727 CD5A07   RBND1:    CALL PLAY     ;CHECK PADDLE
072A 7A                 MOV  A,D      ;POINT LOSS REGISTER
072B FEFF               CPI  PNTLS    ;POINT LOSS CODE
072D C20C07             JNZ  RBND     ;IF NO LOST POINT, REVERSE BALL
0730 0E28               MVI  C,LOSTN  ;LOSE POINT TRILL CODE
0732 CDB907             CALL TRIL     ;PLAY TRILL
0735 C3C806             JMP  SERV     ;NEW SERVE
                        ;
                        ;PASSES BALL FROM ONE PLAYER TO THE OTHER
                        ;
0738 CD9702   PASS:     CALL BLNK     ;CLEAR THE DISPLAY
073B 260B               MVI  H,0BH    ;MSBYTE DISP MEM
073D 160E               MVI  D,0EH    ;BALL PATTERN LENGTH
073F 0A       PASS1:    LDAX B        ;BALL POSITION DISPLAY CODE
0740 5F                 MOV  E,A      ;SAVE A
0741 E612               ANI  12H      ;MASK
0743 0F                 RRC           ;POSITION
0744 6F                 MOV  L,A      ;SAVE
0745 0F                 RRC           ;POSITION
0746 0F                 RRC           ;POSITION
0747 B5                 ORA  L        ;MASK
0748 E603               ANI  3        ;MASK
074A C6FC               ADI  0FCH     ;OFFSET FOR RAM
074C 6F                 MOV  L,A      ;LSBYTE  DISP MEM
074D 7B                 MOV  A,E      ;RESTORE MESSAGE
074E F692               ORI  92H      ;SET DP,E,B SEGS OFF
0750 77                 MOV  M,A      ;SEND TO UDSPX
0751 CDAD07             CALL TIMER    ;DISPLAY IT
0754 03                 INX  B        ;NEXT MESSAGE
0755 15                 DCR  D        ;MESSAGE COUNTER
0756 C23F07             JNZ  PASS1    ;IF LAST BALL POSITION
0759 C9                 RET
                        ;
                        ;PLAY ROUTINE WHEN PASS IS COMPLETE
```

Tabella F-2. Listing della ROM (continuazione)

```
Hex
Adrs Contents Label     Instruction          Comments

                        ;
075A 16FF    PLAY:      MVI  D,PNTLS   ;POINT LOSS CODE
075C CDA207             CALL PADL      ;CHECK PADDLE
075F C8                 RZ             ;IF PUSHED TOO SOON
0760 3A000B             LDA  SPEED     ;PRESENT BALL SPEED
0763 5F                 MOV  E,A       ;SAVE IT
0764 57                 MOV  D,A       ;SET REACTION SPEED COUNTER VALUE
0765 0F                 RRC            ;HALVE BALL SPEED/ TIME
0766 4F                 MOV  C,A       ;SAVE IT
0767 7A      PLAY1:     MOV  A,I       ;RESTORE COUNTER VALUE
0768 3D                 DCR  A         ;COUNTER
0769 B9                 CMP  C         ;HALF TIME
076A CA8207             JZ   FSTR      ;IF PADL NOT PUSHED BY THIS TIME
076D 57                 MOV  D,A       ;SAVE COUNT
076E CDA207             CALL PADL      ;CHECK PADDLE
0771 C26707             JNZ  PLAY1     ;IF NOT PUSHED
0774 0E0B               MVI  C,SLOTN   ;SLOWER TRILL CODE
0776 CDB907             CALL TRIL      ;PLAY TRILL
0779 7B                 MOV  A,E       ;ORIGINAL BALL SPEED
077A FE40               CPI  40H       ;SLOWEST SPEED
077C C8                 RZ             ;IF ALLREADY AT SLOWEST SPEED
077D 07                 RLC            ;HALVE BALL SPEED
077E 32000B             STA  SPEED     ;STORE IT
0781 C9                 RET
                        ;
                        ;FASTER INCREASES RETURN SPEED
                        ;
0782 7A      FSTR:      MOV  A,I       ;RESTORE COUNTER VALUE
0783 16FF               MVI  D,PNTLS   ;LOSE SIGNAL CODE
0785 3D                 DCR  A         ;COUNTER
0786 C8                 RZ             ;IF PADDLE NOT PUSHED IN TIME
0787 57                 MOV  D,A       ;SAVE A
0788 CDA207             CALL PADL      ;CHECK PADDLE
078B C28207             JNZ  FSTR      ;IF NOT PUSHED
078E 0E01               MVI  C,FSTTH   ;FASTER SIGNAL CODE
0790 CDB907             CALL TRIL      ;PLAY TRILL
0793 7B                 MOV  A,E       ;ORIGINAL BALL SPEED
0794 0F                 RRC            ;DOUBLE BALL SPEED
0795 32000B             STA  SPEED     ;STORE IT
0798 C9                 RET
                        ;
                        ;RESET GAME BUTTON PUSHED ?
                        ;
0799 3EFB    RSTGM:     MVI  A,0FBH    ;0 KEY INPUT CODE
079B D328               OUT  SCAN      ;SET SCAN LATCH
079D DB18               IN   KEY       ;INPUT KEYS
079F FEFE               CPI  0FEH      ;0 KEY INPUT CODE
07A1 C9                 RET
                        ;
                        ;PADDLE PUSHED?
                        ;
07A2 CDC801  PADL:      CALL SDS       ;UPDATE DISPLAY
07A5 7B                 MOV  A,E       ;PADDLE KEY SCAN MASK
07A6 D328               OUT  SCAN      ;SET SCAN LATCH
07A8 DB18               IN   KEY       ;INPUT KEYS
07AA FEFB               CPI  0FBH      ;ACCEPTABLE KEY INPUT CODE
07AC C9                 RET
                        ;
                        ;TIME CONTROLS BALL SPEED
                        ;
07AD 3A000B  TIMER:     LDA  SPEED     ;PRESENT BALL SPEED
07B0 CDC801  TIME1:     CALL SDS       ;UPDATE DISPLAY / DELAY
07B3 DE02               SBI  2         ;COUNTER
07B5 C2B007             JNZ  TIME1     ;IF COUNTER NOT DONE
07B8 C9                 RET
                        ;
                        ;TRILL MAKES SPEAKER SOUNDS
                        ;
07B9 D5      TRIL:      PUSH D         ;
07BA C5                 PUSH B         ;
07BB 0606               MVI  B,6       ;START FREQ
07BD 1601               MVI  D,1       ;TONE DURATION
07BF C5      TRIL1:     PUSH B         ;SAVE START FREQ
07C0 CD4704             CALL BEEP2     ;TONE
07C3 C1                 POP  B         ;START FREQ
07C4 04                 INR  B         ;DECREASE FREQ
07C5 79                 MOV  A,C       ;TRILL CODE
07C6 FE01               CPI  FSTTH     ;FASTER CODE
07C8 C2CD07             JNZ  TRIL2     ;IF NOT
07CB 05                 DCR  B         ;RESTORE FREQ
07CC 05                 DCR  B         ;INCREASE FREQ
07CD B8      TRIL2:     CMP  B         ;LAST BEEP
07CE C2BF07             JNZ  TRIL1     ;IF NOT DONE
07D1 C1                 POP  B         ;
07D2 D1                 POP  D         ;
07D3 C9                 RET
                        ;
                        ;LEFT SERVE MESSAGE TABLE
                        ;
07D4 F7      LSM:       DB   0F7H      ;DISPLAY SEGMENT CODES
07D5 F5                 DB   0F5H
07D6 E7                 DB   0E7H
07D7 E5                 DB   0E5H
07D8 E1                 DB   0E1H
07D9 A1                 DB   0A1H
07DA A7                 DB   0A7H
07DB B5                 DB   0B5H
07DC B7                 DB   0B7H
07DD 97                 DB   097H
07DE 96                 DB   096H
07DF B4                 DB   0B4H
07E0 A6                 DB   0A6H
07E1 A0                 DB   0A0H
                        ;
                        ;RIGHT SERVE MESSAGE TABLE
                        ;
07E2 EC      RSM:       DB   0ECH      ;DISPLAY SEGMENT CODES
07E3 EE                 DB   0EEH
07E4 FC                 DB   0FCH
07E5 FE                 DB   0FEH
07E6 DE                 DB   0DEH
07E7 9E                 DB   09EH
07E8 BC                 DB   0BCH
07E9 AE                 DB   0AEH
07EA AC                 DB   0ACH
07EB A8                 DB   0A8H
07EC A0                 DB   0A0H
07ED A6                 DB   0A6H
07EE B4                 DB   0B4H
07EF 96                 DB   096H
0B00         SPEED"     EQU  0B00H
00FF         PNTLS      EQU  0FFH
00FB         LEFT       EQU  0FBH
00F7         RIGHT      EQU  0F7H
0028         LOSTN      EQU  028H
000B         SLOTN      EQU  00BH
0001         FSTTN      EQU  001H
                        ;
                        ;LESSON 3  COUNTER PROGRAM
                        ;
07F0 3E00    XSTART:    MVI  A,0       ;SET A REGISTER TO 0
07F2 3C      XLOOP:     INR  A         ;INCREMENT A REGISTER
07F3 FE0A               CPI  10        ;COMPARE A REGISTER TO 10
07F5 CAF007             JZ   XSTART    ;GO TO BEGINNING IF A=10
07F8 C3F207             JMP  XLOOP     ;INCREMENT AGAIN
07FB 00                 NOP
07FC 00                 NOP
07FD 00                 NOP
07FE 00                 NOP
07FF DE      CHKSM:     DB   0DEH
                        ;
                        ;EQUATES
                        ;
0010         KEY        EQU  10H       ;KEY INPUT PORT ADRS
0020         SIN        EQU  20H       ;KEY INPUT PORT ADRS
0028         SCAN       EQU  28H       ;KEY SCAN PORT ADRS
0030         LOUT       EQU  30H       ;KEY OUTPUT PORT ADRS
0038         DSP        EQU  38H       ;KEY DISPLAY PORT ADRS
000B         DIM        EQU  0BH       ;DEFAULT INTERRUPT MASK
0800         PC         EQU  0800H     ;DEFAULT PROGRAM COUNTER
0BB0         USP        EQU  0BB0H     ;DEFAULT USER STACK POINTER
0BCE         MSP        EQU  0BCEH     ;MONITOR STACK POINTER
008C         TIME       EQU  8CH       ;1MS TIME CONSTANT FOR DELAY LOOP
0006         FREQ       EQU  06H       ;BEEP DEFAULT FREQUENCY CONSTANT
0004         DURA       EQU  04H       ;BEEP DEFAULT DURATION CONSTANT
0AF0         RS4C       EQU  0AF0H     ;RST4   LINK
0AF3         RS5C       EQU  0AF3H     ;RST5   LINK
0AF6         RS55C      EQU  0AF6H     ;RST5.5 LINK
0AF9         RS6C       EQU  0AF9H     ;RST6   LINK
0AFC         RS65C      EQU  0AFCH     ;RST6.5 LINK FOR INTRPT KEY
0AFF         TPR        EQU  0AFFH     ;TOP OF PROTECTED RAM
0AEF         UR         EQU  0AEFH     ;UPPER RAM BELOW RST LINKS
0BD1         TSAVSP     EQU  0BD1H     ;TEMPORARY STACK POINTER SAVE ADRS
0BD3         TSAVH      EQU  0BD3H     ;TEMPORARY H,L REG SAVE ADRS
0BD5         RAML1      EQU  0BD5H     ;RAM LINK TO USER PROG
0BD6         RS         EQU  0BD6H     ;RUN STATUS ADRS
0BD7         RAML2      EQU  0BD7H     ;RAM LINK TO USER PROG
0BDB         SAVIM      EQU  0BDBH     ;SAVE INTERRUPT MASK ADRS
0BDC         SAVPC      EQU  0BDCH     ;SAVE PROGRAM COUNTER ADRS
0BDE         SAVSL      EQU  0BDEH     ;SAVE STACK POINTER LOW ADRS
0BDF         SAVSH      EQU  0BDFH     ;SAVE STACK POINTER HIGH ADRS
0BE0         SAVL       EQU  0BE0H     ;SAVE L REG ADRS
0BE2         SAVE       EQU  0BE2H     ;SAVE E REG ADRS
0BE7         SAVA       EQU  0BE7H     ;SAVE A REG ADRS
0BE8         UDKY       EQU  0BE8H     ;UNDECODED KEY SCAN 0
0BF0         UDSP0      EQU  0BF0H     ;UNDECODED DISPLAY DIGIT 0 ADRS
0BF1         UDSP1      EQU  0BF1H     ;UNDECODED DISPLAY DIGIT 1 ADRS
0BF2         UDSP2      EQU  0BF2H     ;UNDECODED DISPLAY DIGIT 2 ADRS
0BF3         UDSP3      EQU  0BF3H     ;UNDECODED DISPLAY DIGIT 3 ADRS
0BF5         UDSP5      EQU  0BF5H     ;UNDECODED DISPLAY DIGIT 5 ADRS
0BF6         UDSP6      EQU  0BF6H     ;UNDECODED DISPLAY DIGIT 6 ADRS
0BF8         RMP        EQU  0BF8H     ;REGISTER MESSAGE POINTER
0BFA         DDSP0      EQU  0BFAH     ;DECODED DISPLAY DIGIT 0 ADRS
0BFC         DDSP2      EQU  0BFCH     ;DECODED DISPLAY DIGIT 2 ADRS
0BFF         DDSP5      EQU  0BFFH     ;DECODED DISPLAY DIGIT 5 ADRS
0000                    END  0H
```

Tabella F-2. Listing della ROM (continuazione)

APPENDICE G

Espansione del Microprocessor lab

Introduzione

Questa appendice contiene le informazioni necessarie per espandere con delle periferiche il Microprocessor Lab. Nella Tabella G-1 è indicato l'uso ed è data la descrizione generale di tutti i segnali, quelli del bus e quelli speciali. Quando leggete tali descrizioni, fate riferimento allo schema elettrico. Per una spiegazione dettagliata di tutti i segnali del microprocessore 8085 fate riferimento al foglio tecnico (data sheet), riportato nell'Appendice H, e a tutti gli opportuni manuali Intel.

I 48 K indirizzi, che iniziano all'indirizzo 4000 e terminano all'indirizzo FFFF, sono completamente disponibili per una espansione della memoria. Analogamente, utilizzando la linea IO / $\overline{M}$ su P1-S, sono disponibili per una espansione degli I / O gli indirizzi di I / O da 40 a FF.

È possibile accedere ai segnali sui connettori attraverso una coppia di connettori da 44 pin. La TRW Cinch fabbrica dei connettori a 44 pin che rispettano lo spessore da 3 / 32" della scheda del μLab: il loro numero di codice è 251-22-30 341. Potrebbe però essere necessario che si ordinasse un quantitativo superiore ad un certo minimo. Tali connettori possono anche essere acquistati singolarmente presso la Hewlett-Packard con il numero di codice HP 1251-2680.

SEGNALE	LOCAZIONE	UTILIZZO	DESCRIZIONE GENERALE
BUS INDIRIZZI A0-A15	(pin [P1] 1-16)	Espansione del sistema, analisi dell'attività (Analizzatore Logico) e DMA (accesso diretto in memoria) della memoria e dell'I / O presenti sulla scheda.	Tutte le linee degli indirizzi sono demultiplexate e con buffer. Il segnale HLDA fa sì che il bus venga aperto.
BUS DATI D0-D7	(pin [P2] 1-8)	Espansione del sistema, analisi dell'attività e DMA della memoria e dell'I / O presenti sulla scheda.	Collegamento diretto con il bus dei dati del sistema. Se deve essere collegato più di un carico LSTTL, o se viene aggiunta una capacità significativa (>100 pF) dovuta a un cavo o a una scheda, è necessario introdurre un buffer vicino al connettore. Il segnale HLDA fa sì che il bus sia aperto (con resistenze di pull-up da 10 K).
BUS DI INGRESSO DATI D0-D7	(pin [P2] K-7)	Permette di collegare 8 segnali esterni di ingresso alla porta di ingresso del μLab (indirizzo 2000) per un'espansione del sistema.	Gli ingressi sono riportati pari pari alla porta d'ingresso IC13 e agli interruttori d'ingresso S3. Poiché questi interruttori quando diretti verso il basso collegano direttamente a massa tali linee, è necessario che per ogni linea esterna che si vuole controllare il corrispondente interruttore sia rivolto verso l'alto (1). Volendo si possono portare pari pari tutti gli interruttori verso l'alto. Su ogni linea sono poste delle resistenze da 10K. A questa porta si può accedere anche con l'istruzione IN 20.
BUS DI USCITA DATI D0-D7	(pin [P2] 9-16)	Presenta 8 segnali d'uscita memorizzati per una espansione del sistema che utilizzi la porta d'uscitadel μLab (indirizzo3000)	I piedini d'uscita della porta d'uscita IC15 vanno direttamente ai LED da DS12 a DS19 e al connettore. A questa porta si può accedere anche con l'istruzione OUT 30.
INTERRUZIONI RST 5.5 INTR	(P1-L) (P1-K)	Espansione del sistema.	Con RST 5.5 e INTR si danno ulteriori possibilità di interruzione al μLab. Per utilizzare tali linee, è necessario per prima cosa aprire rispettivamente i ponticelli J3 e J2 posti sulla scheda del μLab*.
$\overline{\text{INTA}}$ (Interrupt Acknowledge)	(P1-J)	Gestione di un'interruzione INTR.	Questa uscita del microprocessore risponde al dispositivo che ha generato l'interruzione INTR chiedendo un'ulteriore istruzione.
CLOCK OUT	(P1-D)	Espansione del sistema e sincronizzazione con circuiti esterni.	Un segnale TTL a onda quadra di 2 MHz controllato da un oscillatore a cristallo presente sul μLab.

Tabella G-1 - Segnali sui connettori

SEGNALE	LOCAZIONE	UTILIZZO	DESCRIZIONE GENERALE
VA (Valid Address)	(P1-V)	Espansione del sistema, decodifica degli indirizzi e analisi dell'attività (Analizzatore Logico).	Quando questo segnale è valido, sul bus degli indirizzi è presente un indirizzo valido ed è in corso una lettura o una scrittura in memoria o in I / O.
ALE (Address Latch Enable) $\overline{ALE}$	(P1-A) (P1-U)	Espansione del sistema utilizzando dei chip con i bus dei dati / indirizzi in multiplexer e analisi dell'attività (Analizzatore Logico).	Segnale generato dal microprocessore, che indica che sul bus dei dati è presente un valore d'indirizzo valido. Per comodità dell'utente sono disponibili il segnale vero e il suo negato.
$\overline{WR}$ (Write)	(P1-T)	Espansione del sistema, analisi dell'attività (Analizzatore Logico) e DMA della RAM e delle porte d'uscita presenti sulla scheda.	Uscita con buffer. Il segnale HLDA fa sì che tale linea sia aperta, con DS7A di richiamo. Un circuito esterno deve essere in grado di fornire 4 ma nello stato 0.
$\overline{RD}$ (Read)	(P1-R)	Espansione del sistema, analisi dell'attività (Analizzatore Logico) e DMA della memoria e delle porte d'ingresso presenti sulla scheda.	Uscita con buffer. Il segnale HLDA fa sì che tale linea sia aperta, con DS7B di richiamo. Un circuito esterno deve essere in grado di fornire 4 ma nello stato 0.
IO / $\overline{M}$ (Input-Output / Memory)	(P1-S)	Espansione della memoria o dell'I / O.	Segnale del microprocessore che indica se l'operazione è in memoria o in I / O.
SID (Serial Input Data)	(P1-N)	Espansione del sistema.	Una porta d'ingresso al microprocessore formata da un solo bit, controllata dalla istruzione RIM. Per usare tale ingresso è necessario prima aprire il ponticello J4 sulla scheda del μLab.
SOD (Serial Output Data)	(P1-W)	Espansione del sistema.	Un bit di uscita seriale, il cui dato è identico a quello inviato all'altoparlante ma disaccoppiato. Controllato dall'istruzione SIM. Il segnale HLDA fa sì che tale linea diventi aperta.
READY	(P1-B)	Espansione del sistema con dispositivi lenti sul bus.	È di norma utilizzato dal μLab per effettuare la funzione di single-step. Un livello logico basso su tale linea forza il microprocessore ad attendere i dispositivi lenti posti sul bus. Per utilizzare tale ingresso bisogna aprire il ponticello J5 sulla scheda del μLab. Se si desidera avere anche la funzione single-step, il segnale di step deve essere esternamente riportato indietro alla linea READY (si veda la descrizione della linea STEP data di seguito).

Tabella G-1 - Segnali sui connettori (continuazione)

SEGNALE	LOCAZIONE	UTILIZZO	DESCRIZIONE GENERALE
STEP	(P1-X)	Mantiene la funzione del single-step del μLab per memorie ed I / O esterne ed interne, quando viene utilizzato il segnale esterno di READY.	Quando il ponticello J5 viene aperto, per far sì che una linea esterna possa controllare l'ingresso di READY, i circuiti di single-step del μLab non sono più collegati al microprocessore. Per ripristinare la possibilità del single-step, si può utilizzare il circuito indicato nella Figura G-1.
$\overline{\text{RESET IN}}$	(P1-C)	Controllo esterno del reset all'accensione del μLab.	Un livello basso su questa linea fa sì che il μLab esegua lo stesso ciclo di reset all'accensione che viene effettuato quando viene data l'alimentazione.
RESET OUT	(P1-M)	Inizializzare dei circuiti esterni.	Questa linea si porta a 1 tutte le volte che il microprocessore viene resettato (durante l'accensione o quando la linea $\overline{\text{RESET IN}}$ viene portata bassa).
HOLD	(P1-E)	DMA della memoria e dell'I / O del μLab	Un livello alto su tale ingresso fa sì che il microprocessore entri nello stato di hold. In tale stato la linea HLDA sale alta e forza così gli indirizzi, i dati e le linee di controllo nello stato ad alta impedenza (si veda la descrizione di HLDA). Per utilizzare questo ingresso bisogna prima aprire il ponticello J1 posto sulla scheda del μLab.
HLDA (Hold Acknowledge)	(P1-P)	Disabilita tutte le linee di controllo I / O e memoria del sistema affinché un controllore esterno possa effettuare il DMA.	Quando questa linea è alta, il microprocessore del μLab si trova nello stato di hold e le linee dei bus degli indirizzi, dei dati e di controllo ($\overline{\text{RD}}$, $\overline{\text{WR}}$, IO / $\overline{\text{M}}$) sono nello stato ad alta impedenza. In questa situazione un controllore esterno può trasferire direttamente i dati dai o ai dispositivi sul bus del sistema. HLDA è generato dal segnale HOLD d'ingresso al microprocessore.
S0 (Stato 0) S1 (Stato 1)	(P1-H) (P1-F)	Espansione del sistema	Segnali d'uscita provenienti direttamente dal microprocessore che presenta così delle informazioni di sistema e di stato anticipate.

Tabella G-1 - Segnali sui connettori (continuazione)

* NOTA: I ponticelli (da J1 a J5) sono previsti sul μLab per l'espansione del sistema. Tutti e cinque sono già cortocircuitati dal costruttore utilizzando le piste sulla scheda a circuito stampato. Per aprire un ponticello si de-

ve utilizzare un coltellino affilato in modo da interrompere il sottile collegamento che unisce la coppia di fori passanti che formano il ponticello. Per ripristinare il ponticello si può fare un collegamento tra i due fori o con dello stagno o con un pezzo di filo.

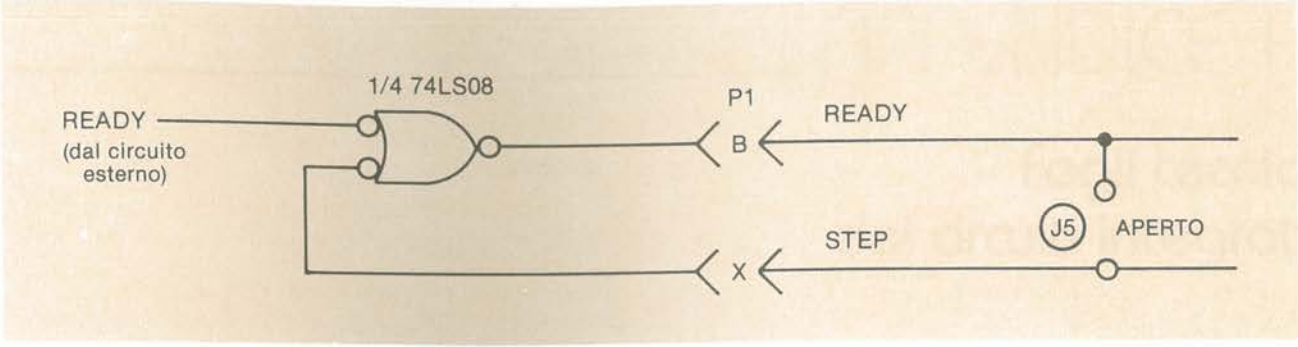


Figura G-1. Un circuito che permette di operare in Single-Step pur utilizzando un ingresso di READY esterno.

COLLEGAMENTI DELLE RESTART

Poiché l'8085 fissa i punti di ingresso delle Restart (RST) e delle interruzioni in una parte di memoria che è occupata dall ROM del μLab, per rilocare in indirizzi della RAM i punti d'ingresso disponibili all'utente sono utilizzati dei collegamenti (link) per i restart. Ad esempio la istruzione di restart RST5 (codice EF) fa sì che si abbia una chiamata all'indirizzo 0028. In 0028 è presente un'istruzione di salto all'indirizzo 0AF3 della RAM del μLab. L'utente poi inserirà una istruzione di salto, a partire dall'indirizzo 0AF3, che farà effettuare al microprocessore un salto ad un altro indirizzo. Quest'ultimo indirizzo conterrà la subroutine voluta e terminerà con un'istruzione di ritorno.

Indirizzo RAM	Label	Collegamento Restart
0AEF	UR	Upper RAM. (Parte superiore della RAM). Istruzione RST7 (FF)
0AF0 0AF1 0AF2	RST4C	Salto alla routine utente di RST4
0AF3 0AF4 0AF5	RST5C	Salto alla routine utente di RST5
0AF6 0AF7 0AF8	RS55C	Salto alla routine utente per l'interruzione RST5.5.
0AF9 0AFA 0AFB	RST6C	Salto alla routine utente di RST6
0AFC 0AFD 0AFE	RS65C	Salto alla routine utente per l'interruzione RST6.5, tasto INTRPT
0AFF	TPR	Top of Protected RAM (termine della RAM protetta). Istruzione RST7 (FF)

Tabella G-2 - Collegamenti delle istruzioni di Restart

APPENDICE H

Fogli tecnici dei circuiti integrati

Questa appendice contiene le specifiche funzionali del microprocessore, della ROM e della RAM, tratte dai fogli tecnici (data sheets) dei fabbricanti. I fogli tecnici degli altri dispositivi presenti nel Microprocessor Lab possono essere trovati nei seguenti manuali (Data Book)

74LS00 74LS14 74LS32 74LS74 74LS138 74LS175 74LS273 74LS374	Texas Instruments TTL Data Book
81LS95 81LS97	National Semiconductor TTL Data Book
8871 75492	National Semiconductor Interface Data Book

intel®

8085A

SINGLE CHIP 8-BIT N-CHANNEL MICROPROCESSOR

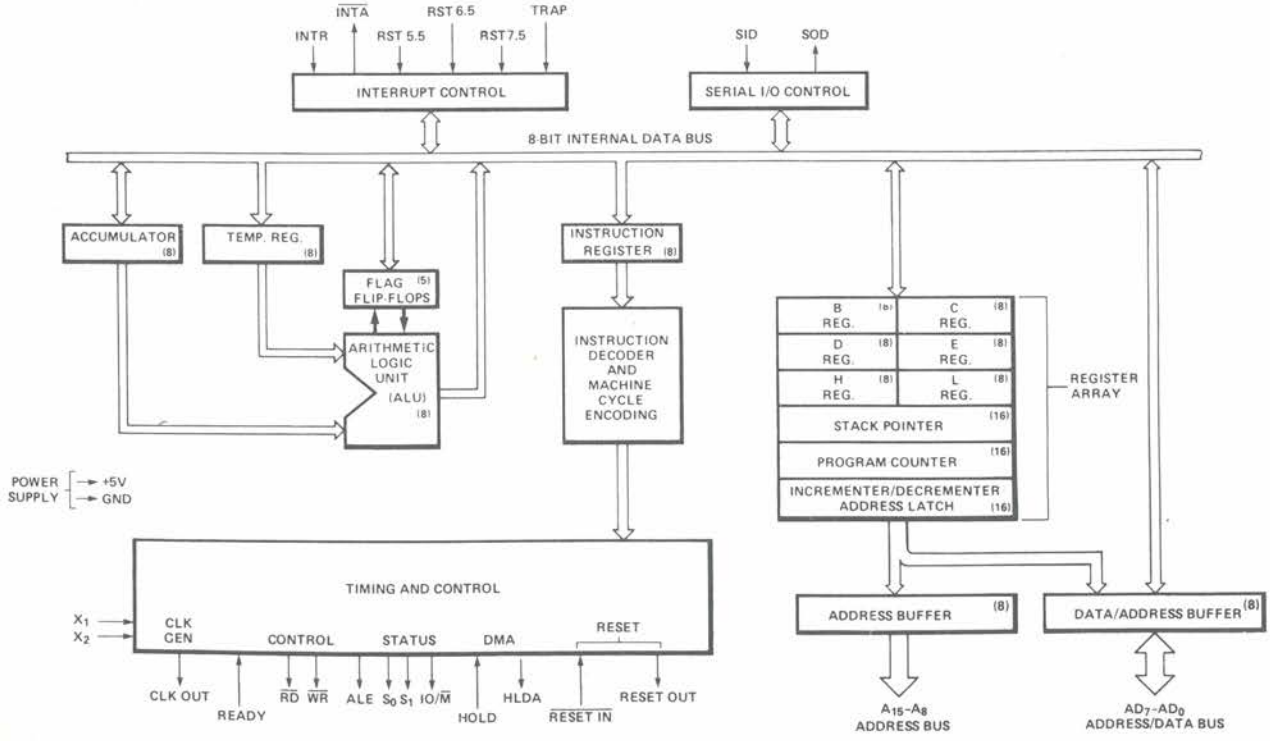
- **Single +5V Supply**
- **100% Software Compatible with 8080A**
- **1.3 μs Instruction Cycle**
- **On-Chip Clock Generator (with External Crystal or RC Network)**
- **On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control**
- **4 Vectored Interrupts (One is Non-Maskable)**
- **Serial In/Serial Out Port**
- **Decimal, Binary, and Double Precision Arithmetic**
- **Direct Addressing Capability to 64K Bytes of Memory**

The Intel® 8085A is a new generation, complete 8 bit parallel central processing unit (CPU). Its instruction set is 100% software compatible with the 8080A microprocessor, and it is designed to improve the present 8080's performance by higher system speed. Its high level of system integration allows a minimum system of 3 IC's: 8085A (CPU), 8156 (RAM), and 8355/8755A (ROM/PROM).

The 8085A incorporates all of the features that the 8224 (clock generator) and 8228 (system controller) provided for the 8080, thereby offering a high level of system integration.

The 8085A uses a multiplexed data bus. The address is split between the 8-bit address bus and the 8-bit data bus. The on-chip address latches of 8155/8156/8355/8755A memory products allows a direct interface with the 8085A.

BLOCK DIAGRAM



Reprinted by Permission of Intel Corporation
Copyright 1977, 1978

PIN DESCRIPTION

The following describes the function of each pin:

A_8-A_{15} (Output 3-State)

Address Bus; The most significant 8-bits of the memory address or the 8-bits of the I/O address, 3-stated during Hold and Halt modes.

AD_{0-7} (Input/Output 3-state)

Multiplexed Address/Data Bus; Lower 8-bits of the memory address (or I/O address) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles.

3-stated during Hold and Halt modes.

ALE (Output)

Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. ALE can also be used to strobe the status information. ALE is never 3-stated.

S_0, S_1 (Output)

Data Bus Status. Encoded status of the bus cycle:

S_1	S_0	
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

S_1 can be used as an advanced R/$\overline{W}$ status.

$\overline{RD}$ (Output 3-state)

READ; indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer. 3-stated during Hold and Halt.

$\overline{WR}$ (Output 3-state)

WRITE; indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes.

READY (Input)

If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input)

HOLD; indicates that another Master is requesting the use of the Address and Data Buses. The CPU, upon receiving the Hold request, will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.

HLDA (Output)

HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

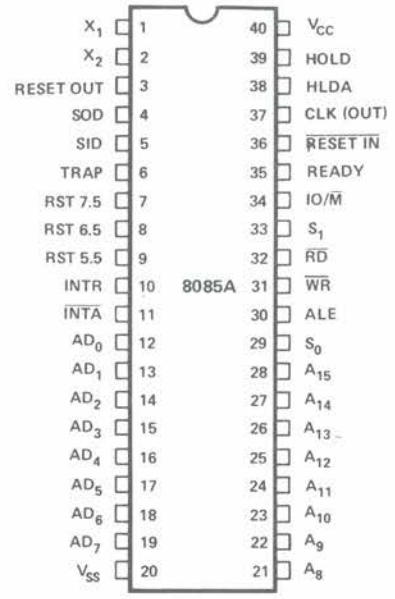


Figure 1. Pin Configuration

INTR (Input)

INTERRUPT REQUEST; is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

$\overline{INTA}$ (Output)

INTERRUPT ACKNOWLEDGE; is used instead of (and has the same timing as) $\overline{RD}$ during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RST 5.5
RST 6.5
RST 7.5
(Inputs)

RESTART INTERRUPTS; These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted.

RST 7.5 → Highest Priority
RST 6.5
RST 5.5 → Lowest Priority

The priority of these interrupts is ordered as shown above. These interrupts have a higher priority than the INTR.

Reprinted by Permission of Intel Corporation
Copyright 1977, 1978

TRAP (Input)

Trap interrupt is a nonmaskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

$\overline{\text{RESET IN}}$ (Input)

Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. None of the other flags or registers (except the instruction register) are affected. The CPU is held in the reset condition as long as Reset is applied.

RESET OUT (Output)

Indicates CPU is being reset. Can be used as a system RESET. The signal is synchronized to the processor clock.

X_1, X_2 (Input)

Crystal or R/C network connections to set the internal clock generator. X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

CLK (Output)

Clock Output for use as a system clock when a crystal or R/C network is used as an input to the CPU. The period of CLK is twice the X_1, X_2 input period.

IO/$\overline{\text{M}}$ (Output)

IO/$\overline{\text{M}}$ indicates whether the Read/Write is to memory or I/O. Tri-stated during Hold and Halt modes.

SID (Input)

Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output)

Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

V_{CC}

+5 volt supply.

V_{SS}

Ground Reference.

Reprinted by Permission of Intel Corporation
Copyright 1977, 1978

2048x8 Static Read Only Memory

SY2316A SY2316B

Synertek®

MEMORY PRODUCTS

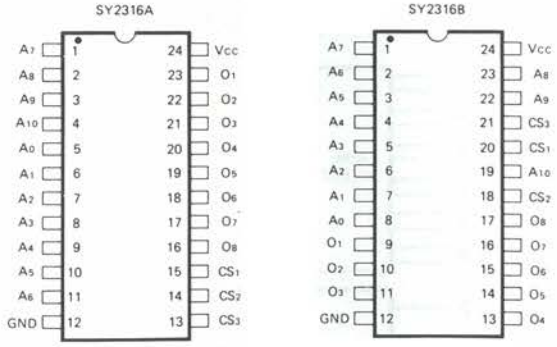
- 2048x8 Bit Organization
- Single +5 Volt Supply
- Metal Mask Programming
- Two Week Prototype Turnaround
- Access Time—550ns /450ns (max.)
- Totally Static Operation
- Completely TTL Compatible
- Three-State Outputs for Wire-OR Expansion
- Three Programmable Chip Selects
- SY2316A – Replacement for Intel 2316A
- SY2316B – Pin Compatible with 2708 EPROM
  – Replacement for Two 2708s

The SY2316A and SY2316B high performance read only memories are organized 2048 words by 8 bits with access times of less than 550 ns and 450 ns. These ROMs are designed to be compatible with all microprocessor and similar applications where high performance, large bit storage and simple interfacing are important design considerations. These devices offer TTL input and output levels with a minimum of 0.4 Volt noise immunity in conjunction with a +5 Volt power supply.

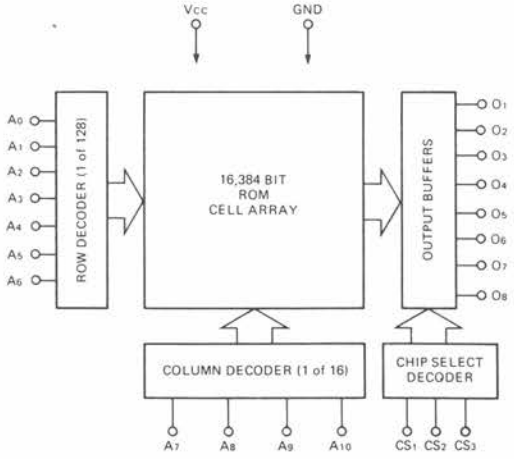
The SY2316A/B operate totally asynchronously. No clock input is required. The three programmable Chip Select inputs allow eight 16K ROMs to be OR-tied without external decoding. Both devices offer three-state output buffers for memory expansion.

Designed to replace two 2708 8K EPROMs, the SY2316B can eliminate the need to redesign printed circuit boards for volume mask programmed ROMs after prototyping with EPROMs.

PIN CONFIGURATION



BLOCK DIAGRAM



ORDERING INFORMATION

Order Number	Package Type	Access Time	Temperature Range
SYC2316A	Ceramic	550ns	0°C to +70°C
SYP2316A	Plastic	550ns	0°C to +70°C
SYC2316B	Ceramic	450ns	0°C to +70°C
SYP2316B	Plastic	450ns	0°C to +70°C

A custom number will be assigned by Synertek.

Reprinted by Permission of Synertek Incorporated
Copyright 1976.

2114
1024 X 4 BIT STATIC RAM

	2114-2	2114-3	2114	2114L2	2114L3	2114L
Max. Access Time (ns)	200	300	450	200	300	450
Max. Power Dissipation (mw)	525	525	525	370	370	370

- **High Density 18 Pin Package**
- **Identical Cycle and Access Times**
- **Single +5V Supply**
- **No Clock or Timing Strobe Required**
- **Completely Static Memory**
- **Directly TTL Compatible: All Inputs and Outputs**
- **Common Data Input and Output Using Three-State Outputs**
- **Pin-Out Compatible with 3605 and 3625 Bipolar PROMs**

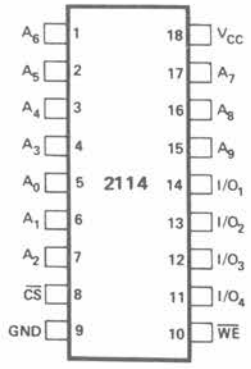
The Intel® 2114 is a 4096-bit static Random Access Memory organized as 1024 words by 4-bits using N-channel Silicon-Gate MOS technology. It uses fully DC stable (static) circuitry throughout — in both the array and the decoding — and therefore requires no clocks or refreshing to operate. Data access is particularly simple since address setup times are not required. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

The 2114 is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives. The 2114 is placed in an 18-pin package for the highest possible density.

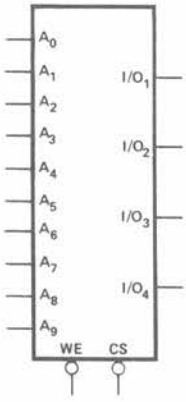
It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. A separate Chip Select ($\overline{CS}$) lead allows easy selection of an individual package when outputs are or-tied.

The 2114 is fabricated with Intel's N-channel Silicon-Gate technology — a technology providing excellent protection against contamination permitting the use of low cost plastic packaging.

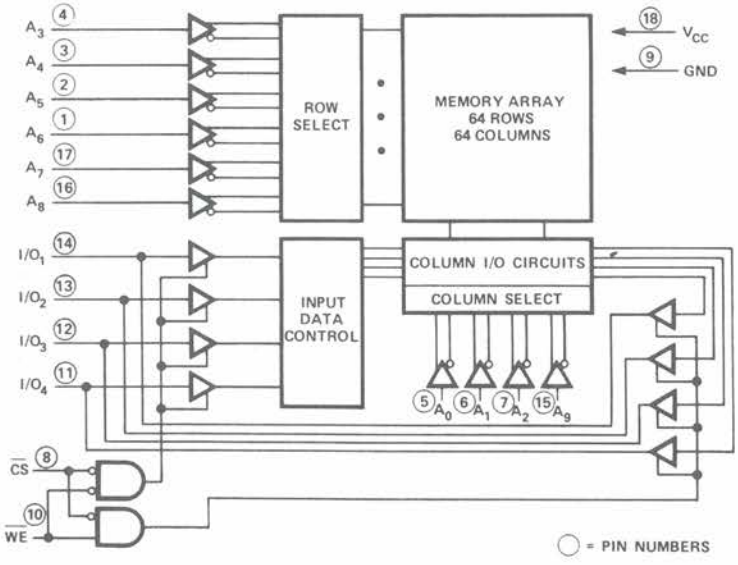
PIN CONFIGURATION



LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

A_0–A_9	ADDRESS INPUTS	V_{CC} POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND GROUND
$\overline{CS}$	CHIP SELECT	
I/O_1–I/O_4	DATA INPUT/OUTPUT	

Reprinted by Permission of Intel Corporation
Copyright 1977, 1978

GLOSSARIO

GLOSSARIO

A

ABILITAZIONE (Enable)
Segnale d'ingresso che permette ad un dispositivo di svolgere la funzione che gli è propria.

ABORTO (Abort)
Ferma il programma e restituisce il controllo all'operatore o al sistema operativo. Nel μLab tale operazione è effettuata dal tasto RESET.

ACCESSO CASUALE (Random Access)
Metodo di accesso per cui per ricuperare ogni parola è necessario sempre lo stesso tempo. Alle locazioni di memoria si può accedere cioè in qualunque (casuale) ordine.

ACCESSO CASUALE IN MEMORIA
Si veda DMA.

ACCESSO, TEMPO DI (Access Time)
Il tempo richiesto per ricevere un dato valido da un dispositivo di memoria a partire da un segnale di lettura.

ACCUMULATORE (Accumulator)
Uno o più registri, associati all'Unità Aritmetico-Logica (ALU), che memorizzano in modo temporaneo somme e altri risultati aritmetici e logici della ALU stessa.

A/D
Si veda: ANALOGICO-DIGITALE, CONVERTITORE.

ADC
Si veda: ANALOGICO-DIGITALE, CONVERTITORE.

ALFANUMERICO (Alphanumeric)
Un carattere che può essere alfabetico o numerico.

ALGORITMO (Algorithm)
Procedimento formato da una sequenza di passi per la soluzione di un problema. Per prima cosa viene identificato il problema, quindi viene trovato un algoritmo che ne dà la soluzione.

ALU
Si veda: UNITA' ARITMETICO-LOGICA.

ANALISI DELLA FIRMA (Signature Analysis)
Tecnica utilizzata per facilitare la ricerca dei guasti nei circuiti digitali. I nodi di un circuito, stimolati opportunamente all'interno di un modo di test, producono delle «firme» come risultato del processo di compressione dei dati effettuato dall'Analizzatore di firma. È possibile identificare dei nodi guasti, confrontando le firme rilevate sui nodi con quelle conosciute come corrette e riportate sulla documentazione.

ANALIZZATORE DI FIRMA (Signature Analyzer)
Strumento che serve a convertire in firme di quattro cifre le lunghe e complesse sequenze seriali di dati che sono presenti sui nodi di un sistema a microprocessore.

ANALIZZATORE LOGICO (Logic Analyzer)
Sistema di un test in grado di visualizzare 0 e 1 e di effettuare delle complesse operazioni di test. Gli analizzatori logici hanno tipicamente da 16 a 32 linee di ingresso e, per ciascuna linea d'ingresso, possono memorizzare sequenze di sedici o più bit.

ANALOGICO (Analog)
Riferito a segnali che abbiano un intervallo continuo di valori di tensione o di corrente.

ANALOGICO-DIGITALE, CONVERTITORE (Analog to Digital Converter)
Converte tensioni e correnti analogiche nella rappresentazione digitale utilizzata dai calcolatori, che possono così «sentire» segnali generati dal mondo «reale».

ANELLO APERTO (Open Loop)
Circuito che opera senza reazione.

ANELLO CHIUSO (Closed Loop)
Circuito che opera all'interno di una reazione, e i cui ingressi sono funzione delle sue uscite.

ANTIRIMBALZO (Debounce)
Processo per eliminare i rimbalzi, generati dagli interruttori meccanici, sui segnali. L'antirimbalzo può essere realizzato sia da hardware che da software.

ARCHITETTURA (Architecture)
Struttura logica di un calcolatore.

ASCII
American Standard Code for Information Interchange, cioè Codice Americano Normalizzato per lo Scambio di Informazioni. Codice per i caratteri utilizzato nella maggior parte dei calcolatori per rappresentare le informazioni.

ASCOLTATORE (Listener)
Dispositivo che preleva dei dati dal bus dei dati. Una porta d'uscita è un dispositivo ascoltatore.

ASINCRONO (Asynchronous)
Un qualunque sistema o circuito che non sia sincronizzato da un segnale di clock comune.

ASSEMBLAGGIO, LINGUAGGIO DI (Assembly Language)
Linguaggio orientato al calcolatore. Un programma è di solito formato da una sequenza di istruzioni che fanno uso di simboli mnemonici che suggeriscono la funzione dell'istruzione. Viene quindi tradotto dal programma assembler in linguaggio macchina.

ASSEMBLARE A MANO (Hand Assemble)
Tradurre un programma da linguaggio di assemblaggio a codice macchina senza l'ausilio di un programma assemblatore (assembler).

ASSEMBLER, PROGRAMMA (Assembler Program)
Traduce in linguaggio macchina le istruzioni scritte in linguaggio di assemblaggio (mnemonici). È anche detto programma assemblatore.

AUTOTEST (Self-Test)
Test che un prodotto effettua su se stesso.

AZZERAMENTO (Clear)
Portare un circuito in uno stato conosciuto, di solito zero.

B

BACKPLANE
La scheda o il pannello in cui tutte le altre schede si inseriscono. Di solito contiene i bus del sistema. Talvolta è anche detta Motherboard.

BAFFI DI STAGNO (Solder Bridge)
Eccedenze di stagno che mettono in corto due conduttori. Un problema comune nella produzione dei circuiti stampati.

BASE (Base)
Si veda: RADICE.

BASIC
Un linguaggio facile da apprendere e facile da utilizzare che è disponibile sulla maggior parte dei microcalcolatori.

BAUD RATE
Misura del flusso dei dati: il numero di unità di segnale al secondo. Quando una unità corrisponde a un bit, il Baud rate (lett.: la frequenza di Baud) è uguale numericamente al numero di bit per secondo (bps). Le telescriventi (teletype, TTY) trasmettono a 110 baud. Ogni carattere è formato da 11 bit, e la telescrivente trasmetterà così 10 caratteri al secondo.

BCD
Binary Coded Decimal, cioè Decimale Codificato in Binario. Una rappresentazione in 4 bit delle 10 cifre decimali da «0» a «9». Dei sedici codici possibili sei non vengono utilizzati. Di solito in un byte vengono «impaccate» due cifre BCD.

BENCHMARK
Metodo per misurare in ben preciso ambito le prestazioni di un calcolatore.

BIDIREZIONALE (Bidirectional)
Indica che il flusso del segnale può avvenire in entrambe le direzioni. I comuni bus bidirezionali sono TTL: three-state o open collector.

BINARIO (Binary)
Un sistema di numerazione che utilizza 2 come base, a differenza del sistema decimale che usa la base 10. Il sistema binario fa uso di due soli simboli, 0 e 1. In binario il valore 2 è espresso con il numero 10.

BINARY SEARCH
Si veda: RICERCA BINARIA.

BIT
Contrazione di Binary digit (cifra binaria). Una cifra in un numero binario.

BIT MENO SIGNIFICATIVO (LSD) (Least Significant Bit)
All'interno di una parola binaria è il bit posto all'estrema destra ed ha il peso numerico più basso.

BIT PIÙ SIGNIFICATIVO (MSB) (Most Significant Bit)
All'interno di una parola binaria è il bit posto all'estrema sinistra ed ha il peso numerico più elevato.

BIT SLICE
Metodo per cui la CPU viene suddivisa in fette (slice) di un bit ciascuna. Ogni chip di un processore bit-slice contiene un percorso completo per i dati attraverso la CPU. Un processore a 32 bit può essere costruito utilizzando otto «fette» di CPU di quattro bit ciascuna.

BOLLE, MEMORIA A (Bubble Memory)
Memoria che utilizza microscopici domini magnetici su di un substrato di granato alluminoso. Sono importanti in quanto sono memorie a stato solido non volatili.

BOOLEANA, LOGICA (Boolean Logic)
Così detta in onore di George Boole, che definì l'aritmetica binaria e le operazioni logiche, tra cui AND, OR, NOT e XOR.

BOOTSTRAP
Programma utilizzato per inizializzare il calcolatore. Di solito azzera la memoria, prepara i dispositivi di I/O e carica il sistema operativo.

BRANCH
Deviazione. Si veda: SALTO.

BREAKPOINT
Punto di interruzione. Meccanismo hardware o software che, al verificarsi di condizioni definite dall'utente, ferma il programma e salva lo stato corrente della macchina.

BUFFER
Un circuito integrato che è utilizzato per ripristinare il livello logico corretto.

BUG
Baco, errore. Quando si eliminano gli errori si dice che si sta facendo un'operazione di debug o, in gergo, che si sta «debuggando».

BURN-IN
È un metodo di test dei componenti che è utilizzato per evidenziare subito all'inizio dei guasti facendo funzionare il circuito per un certo lasso di tempo.

BUS
Percorso per segnali che hanno una funzione comune. La maggior parte dei microprocessori utilizza tre bus: il bus dei dati, il bus degli indirizzi e il bus di controllo.

BUS, CONFLITTO SUL (Bus Conflict)
Situazione anomala che si verifica quando due o più uscite di dispositivi, poste a stati logici opposti, sono presentate contemporaneamente su un bus a tre stati.

BUS, CONTROLLORE DEL (Bus Controller)
Genera i comandi del bus e i segnali di controllo.

BUS DRIVER
Un circuito integrato che viene posto su di un bus affinché i segnali generati dalla CPU siano in grado di pilotare gli altri dispositivi collegati al bus.
Questo è necessario a causa della presenza dei carichi capacitivi, che rallentano i segnali dei dati e possono compromettere la sequenza di temporizzazioni proprie dell'operazione in corso.

BUS, TERMINAZIONE DEL (Bus Termination)
Metodo per impedire che si generino riflessioni all'estremità di un bus. È necessaria solo in sistemi molto veloci.

BYTE
Gruppo di 8 bit. Può essere utilizzato per rappresentare un carattere. Le istruzioni dei microcalcolatori richiedono uno, due o tre byte. Una parola può essere formata da uno o più byte.

C

CALCOLATORE (Computer)
Sistema di calcolo di utilizzo generale che è formato dalla CPU, dalla memoria, dalle unità di I/O e dall'alimentatore.

CALL
Chiamata di una subroutine. Viene effettuato un salto all'indirizzo indicato, ma viene anche salvato il contenuto del Program Counter (di solito nello stack). In tal modo, quando la subroutine è terminata, può essere ripreso il flusso del programma che ha effettuato la chiamata.

CARATTERI, GENERATORE DI (Character Generator)
Circuito che forma le lettere o le cifre su un display o su una stampante.

CARRY, FLAG DI (Carry Flag)
Bit di indicazione, presente nel registro di stato del microprocessore, che è utilizzato per segnalare l'overflow di una operazione effettuata dall'Unità Aritmetico-Logica.

CCD
Charge Compled Device, cioè Dispositivo ad Accoppiamento di Carica. Tecnologia di memorizzazione in seriale che fa uso di condensatori MOS.

CHECKERBOARD
Metodo di test della memoria in cui le celle della memoria sono caricate con 0 e 1 alternati.

CHECKSUM
Somma di controllo. Metodo utilizzato per verificare la correttezza dei dati caricati in un calcolatore.

CHIAMATA
Si veda CALL.

CHIP
Nome diffuso con cui si indicano tutti i circuiti integrati.

CHIP ENABLE (CE)
Abilitazione del chip. Si veda CHIP SELECT.

CHIP SELECT (CS)
Selezione del chip. Abilita di solito i driver a tre stati sulle linee di uscita del chip. La maggior parte dei chip LSI hanno una o più linee di selezione del chip. La linea CS è utilizzata per selezionare uno solo tra più chip.

CICLO MACCHINA (Machine Cycle)
In un sistema, il periodo base di tempo richiesto per manipolare un dato.

CICLO, TEMPO DI (Cycle Time)
Il tempo complessivo richiesto da un dispositivo di memoria per completare un ciclo di lettura o di scrittura ed essere di nuovo pronto per una operazione successiva.

CIFRA (Digit)
Segno o simbolo utilizzato da solo o unito ad altri numeri dello stesso insieme, per dare una certa quantità di informazioni: 2, 3, 4 e 5 sono delle cifre. Deve essere specificata la base, o radice, e deve essere assegnato il valore di ogni cifra.

CLOCK
Generatore del tempo di riferimento di un sistema. Un clock (orologio) genera degli impulsi regolari che sincronizzano o «agganciano» gli eventi.

CODICE (Code)
Il linguaggio macchina stesso.

CODICE MACCHINA (Machine Code)
Si veda LINGUAGGIO MACCHINA.

CODICE MNEMONICO (Mnemonic Code)
Codici introdotti come aiuto per la memoria umana. Il linguaggio dei microprocessori è costituito da parole binarie, cioè formate da una sequenza di 0 e 1, e non sarebbe quindi per niente facile ricordare le istruzioni corrispondenti ad una certa operazione. Perché sia più facile ricordarli, ai codici binari vengono associati gruppi di lettere (detti simboli mnemonici) che suggeriscono, in un certo modo, la definizione della istruzione. Così il codice 32 per il microprocessore 8085 significa «carica l'accumulatore» (load accumulator) e viene rappresentato con il simbolo mnemonico LED.

CODICE OPERATIVO (OPCODE) (Operation Code)
Parte di ogni istruzione, espressa in linguaggio macchina, che specifica quale operazione deve essere effettuata. Le altre parti specificano il dato, l'indirizzo o la porta. Nel caso dell'8085, il primo byte di ogni istruzione è il codice operativo.

CODICE SORGENTE (Source Code)
Programma scritto in un linguaggio che non sia il linguaggio macchina. Può essere un linguaggio di assemblaggio o un linguaggio ad alto livello.

CODIFICARE (Code)
Il processo di conversione da un linguaggio ad un altro.

COMMENTI, CAMPO (Comment Field)
All'interno di un'istruzione è il campo riservato ai commenti. Viene ignorato dal compilatore o dall'assembler, quando il programma viene convertito in codice macchina.

COMPARATORE LOGICO (Logic Comparator)
Strumento di test che confronta, per ogni piedino, il comportamento di un dispositivo inserito nel circuito sotto esame con il comportamento di un uguale dispositivo funzionante di riferimento.

COMPILATORE (Compiler)
Programma di traduzione che converte delle istruzioni ad alto livello in gruppi di istruzioni (in codice macchina) eseguibili. Tutti i linguaggi ad alto livello richiedono un compilatore o un interprete. Un compilatore traduce il programma completo che viene successivamente eseguito.

COMPLEMENTARE (Complement)
Processo che cambia gli 0 in 1 e gli 1 in 0.

COMPLEMENTO A DUE (Two's Complement)
Sistema di numerazione utilizzato per rappresentare dei numeri sia positivi che negativi. Nella rappresentazione in complemento a due i numeri positivi coincidono con i numeri positivi del sistema di numerazione binario standard. La rappresentazione in complemento a due di un numero negativo è invece il complemento del valore assoluto espresso in binario, più 1. Si noti che l'ottavo bit (o in generale il più significativo) indica il segno: 0 = positivo, 1 = negativo.

COMPLEMENTO A UNO (One's Complement)
Sistema di rappresentazione numerica utilizzato per numeri interi binari dotati di segno, in cui per ottenere il negativo di un numero è sufficiente complementarlo. Il bit posto all'estrema sinistra diventa il bit di segno, 0 ad indicare positivo, 1 negativo.

CONDIZIONATO, SALTO O CHIAMATA (Conditional Jump or Call)
Istruzione che, quando eseguita, a seconda del risultato di una certa condizione, farà sì che il calcolatore continui con la istruzione del programma o trasferisca il controllo ad una istruzione diversa.

CONDIZIONE, CODICE DI (Condition Code)
Si riferisce a un limitato numero di condizioni di programma (ad esempio il carry, il riporto negativo, l'overflow, ecc.) che sono legate all'esecuzione delle istruzioni. Questi codici sono contenuti in un registro codici di condizione, detto anche *registro dei flag*.

CONTATORE DI PROGRAMMA
Si veda PROGRAM COUNTER.

CONTROLLO, BUS DI (Control Bus)
Insieme di linee di controllo all'interno di un calcolatore. Servono a sincronizzare e a controllare le informazioni necessarie affinché il sistema possa girare.

CONTROLLO, PROGRAMMA DI (Control Program)
Sequenza di istruzioni che guidano la CPU attraverso le diverse operazioni che essa deve effettuare. Questo programma è memorizzato in modo permanente nella particolare memoria ROM cui la CPU può accedere nel corso delle operazioni. Di solito tale ROM è contenuta all'interno del chip del microprocessore detto anche *microprogramma* o *microcodice*.

CONTROLLO, SEZIONE DI (Control Block)
Circuiti che nella CPU svolgono le funzioni di controllo. Sono i circuiti che decodificano le istruzioni e generano quindi i segnali interni di controllo che realizzano le operazioni richieste.

CORRENTE, RIVELATORE DI (Current Tracer)
Strumento portatile per la ricerca guasti utilizzato per rivelare il flusso di corrente nei circuiti logici.

COSTANTE (Constant)
Un valore fisso.

CPS
Character per Second, cioè Caratteri al Secondo.

CPU
Central Processing Unit, cioè Unità Centrale di Processo. Parte del calcolatore che ha la funzione di prelevare, decodificare ed eseguire le istruzioni. Contiene un'unità di controllo, una ALU e vari accessori (registri, clock, driver).

CRC
Cyclic Redundancy Check, cioè Verifica Ciclica della Ridondanza. Polinomiale binario. È utilizzato per generare delle informazioni di controllo al checksum, ma è più difficile da generare e più affidabile.

CROSS - ASSEMBLER
Assembler che gira su di un processore il cui linguaggio d'assemblaggio è diverso dal linguaggio che viene assemblato.

CROSSTALK
Interferenza tra due segnali.

CRT, TERMINALE (CRT Terminal)
Terminale di un calcolatore che è formato da un display CRT (Cathode Ray Tube, cioè Tubo a Raggi Catodici) e da una tastiera; è di solito collegato al calcolatore attraverso un canale seriale.

CYCLIC REDUNDANCY CHECK
Si veda CRC.

D

D/A
Si veda: DIGITALE-ANALOGICO, CONVERTITORE.

DAC
Si veda: DIGITALE-ANALOGICO, CONVERTITORE.

DAISY CHAIN
Linea del bus di controllo collegata ai dispositivi in modo tale che il segnale passa da un dispositivo al successivo in modo «seriale».

DATO, DATI (Data)
Termine generico che indica uno o più fatti, numeri, lettere e simboli che si riferiscono o descrivono un oggetto, un'idea, una condizione, una situazione o altri fattori. Indica gli elementi fondamentali d'informazione che possono essere elaborati o prodotti da un calcolatore. Talvolta si pensa che i dati possano essere espressi solo in forma numerica, ma in realtà le informazioni non sono essenzialmente numeriche.

DATI, ACQUISIZIONE DEI (Data Acquisition)
Raccolta di dati da sensori esterni, solitamente in forma analogica.

DATA BASE
Organizzazione sistematica di archivi (file) di dati, al fine di permettere un rapido accesso, ricupero e aggiornamento degli stessi.

DATI, BUS DEI (Data Bus)
Insieme di linee su cui vengono trasferiti i dati. Il bus dei dati è solitamente bidirezionale e a tre stati.

DEBUG, EFFETTUARE UN (Debugging)
Procedimento di eliminazione degli errori hardware o software presenti in un sistema.

DEBUGGER
Programma che facilita l'operazione di debug del software. In generale permette di introdurre dei breakpoint, fornisce delle possibilità di dump e la possibilità di esaminare e alterare registri e memoria.

DECODER
Decodificatore. Dispositivo logico che decodifica ingressi binari. Un decoder a 3 bit (p.es. il 74138) avrà $2^3 = 8$ uscite, in quanto un numero di 3 bit può assumere otto valori diversi.

DECODIFICA CON SELEZIONE LINEARE (Linear Select Decoding)
Metodo di decodifica degli indirizzi che per abilitare i dispositivi di un sistema fa uso pari pari dei bit più pesanti degli indirizzi.

DECODIFICA DEGLI INDIRIZZI (Address Decoding)
Operazione attraverso cui un dato indirizzo o campo di indirizzi viene selezionato ad abilitare dei dispositivi.

DECREMENTO (Decrement)
Istruzione che diminuisce il contenuto di una locazione di memoria.

DEDICATO (Dedicated)
Realizzato «ad hoc» per qualche uso particolare.
Un microprocessore dedicato è un microprocessore che è stato programmato in modo particolare per una ben precisa applicazione, ad esempio bilance, semafori, ecc. Le ROM sono, per loro stessa natura, delle memorie dedicate.

DIAGRAMMA DI FLUSSO (Flowchart or Flow Diagram)
Rappresentazione grafica delle funzioni logiche di un programma. I diagrammi di flusso permettono al progettista di visualizzare le procedure necessarie per ogni parte del programma. Un diagramma di flusso completo permette di passare direttamente al codice finale.

DIGIT
Si veda CIFRA.

DIGITALE (Digital)
Che ha degli stati discreti. La maggior parte della logica digitale è binaria, cioè con due stati (on/off, acceso/spento).

DIGITALE-ANALOGICO, CONVERTITORE (Digital to Analog Converter)
Effettua la conversione dalla rappresentazione digitale, utilizzata nei calcolatori, a quella analogica tipica del mondo che ci circonda.

DIGITALIZZARE (Digitize)
Il processo di conversione di una quantità analogica in una quantità digitale.

DIP
Dual In-line Package, cioè Contenitore a Doppia Fila. Contenitore standard per circuiti integrati con due file parallele di piedini.

DISABILITARE (Disable)
Impedire il funzionamento di un dispositivo.

DMA
Direct Memory Access, cioè Accesso Diretto in Memoria. Metodo per ottenere l'accesso diretto alla memoria principale, al fine di effettuare dei trasferimenti di dati senza l'intervento della CPU.

DOMINIO DEI DATI (Data Domain)
Analisi o visualizzazione di segnali di cui venga considerato solo il valore digitale e non il valore della tensione o del tempo precisi. Un analizzatore degli stati logici visualizza le informazioni nel dominio dei dati.

DOMINIO DEL TEMPO (Time Domain)
Informazione che è direttamente legata al tempo. Un oscilloscopio visualizza le informazioni nel dominio del tempo.

DOPPIA PRECISIONE, ARITMETICA IN (Double Precision Arithmetic)
Fa uso di due parole per rappresentare ogni numero.

DOS
Disk Operating System, cioè Sistema Operativo basato su Disco.

DUMP
Trasferire il contenuto di grosse aree di memoria.

E

EAROM
Electrically Alterable Read Only Memory, cioè Memoria a Sola Lettura Alterabile Elettricamente. Una ROM che può essere modificata, senza essere tolta dal circuito, con mezzi elettrici.

ECC
Error Correcting Code, Cioè Codice con Correzione d'Errore. Codice che utilizzando dei bit addizionali permette di rivelare e correggere automaticamente gli errori.

ECO (Echo)
L'azione per cui un carattere ricevuto dalla tastiera viene inviato alla stampante o al display.

EPROM
Erasable Programmable Read Only Memory, cioè Memoria a Sola Lettura Programmabile e Cancellabile. Una PROM che può essere riutilizzata. La maggior parte delle EPROM possono essere cancellate con una esposizione alla luce ultravioletta.

ESADECIMALE (Hexadecimal)
Sistema numerico in base 16. Poichè si hanno 16 cifre esadecimali (da 0 a 15) e solo dieci cifre numeriche (da 0 a 9), sono necessarie altre sei cifre per rappresentare i valori da 10 a 15. A tale scopo vengono usate le prime sei lettere dell'alfabeto. Le cifre esadecimali diventano così: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Il numero decimale 16 diventa così il numero esadecimale 10. Il numero decimale 26 diventa il numero esadecimale 1A.

ESECUZIONE, CICLO DI (Execute Cycle)
L'ultimo ciclo che viene effettuato nel corso dell'esecuzione di una istruzione. Durante tale tempo viene svolta l'operazione propria dell'istruzione.

ESECUZIONE, TEMPO DI (Execution Time)
Tempo richiesto perché una istruzione sia eseguita.

ESPONENTE (Exponent)
Potenza di dieci per cui viene moltiplicato un numero; utilizzato nella rappresentazione a virgola mobile. Ad esempio, l'esponente del numero decimale $0{,}9873 \times 10^7$ è 7.

F

FAN-IN
Carico elettrico presentato da un ingresso. Di solito è espresso come numero di carichi d'ingresso equivalenti standard.

FAN-OUT
Carico elettrico che un'uscita può pilotare. Di solito è espresso come numero di ingressi che possono essere pilotati.

FETCH
Prelievo. La lettura di una istruzione dalla memoria.

FIFO
First-In - First-Out, cioè Primo a Entrare - Primo a Uscire. Struttura di memoria in cui il dato viene inserito da un lato e fatto uscire dall'altro. Un FIFO è utilizzato come un buffer per collegare due dispositivi che operano in modo asincrono.

FIRMA (Signature)
Valoredi quattro cifre, generato da un analizzatore di firma. Viene utilizzato per caratterizzare l'attività dei dati presente su di un nodo logico durante un certo intervallo di tempo.

FIRMWARE
Programma memorizzato in ROM. Di solito per firmware si intende qualunque programma inserito in ROM.

FLAG
Bit d'informazione che indica che è stata raggiunta una qualche condizione limite, come nel caso dell'overflow o del carry. È anche un indicatore di condizioni particolari, ad esempio le interruzioni.

FLOPPY DISK
Dispositivo di memoria di massa che per immagazzinare le informazioni fa uso di un dischetto flessibile (floppy).

FLOWCHART
Si veda DIAGRAMMA DI FLUSSO.

FLUTTUANTE (Floating)
Nodo logico su cui non siano presenti uscite attive. Le linee del bus a tre stati, ad esempio le linee del bus dei dati, sono fluttuanti quando nessun dispositivo è abilitato.

FREE-RUN
Procedimento che permette ad un circuito digitale (tipicamente un microprocessore) di operare senza reazione (ad anello aperto). Viene utilizzato al fine di stimolare gli altri dispositivi del circuito in modo prevedibile e ripetitivo.

FRONTE DI DISCESA (Falling Edge)
Transizione logica da alto a basso.

FRONTE DI SALITA (Rising Edge)
Transizione logica da basso ad alto.

G

GENERATORE DI IMPULSI LOGICI (Logic Pulser)
Strumento portatile di ricerca guasti che immette nei modi logici dei segnali digitali controllati.

GLITCH
Uno o più impulsi di rumore. È anche usato per indicare un qualunque guasto, non spiegato, del sistema.

H

HALF SPLITTING
Tecnica di riparazione utilizzata per isolare i guasti. Implica l'esame dei nodi del circuito posti all'incirca a metà dello stesso. Quando si sia definito lo stato di questi nodi, si potrà associare la causa del guasto ai circuiti che precedono il punto centrale o a quelli che lo seguono.

HALT
Alt. Comando che arresta il calcolatore.

HANDSHAKE
Segnali di controllo di una interfaccia: il dispositivo che sta effettuando l'invio di un dato genera un segnale ad indicare che è disponibile una nuova informazione, mentre il dispositivo che riceve risponde con un altro segnale ad indicare che il dato è stato ricevuto.

HARDWARE
Per lungo tempo nel gergo dei progettisti sono stati indicati con tale nome i singoli componenti, attivi e passivi, di un circuito. Oggi è genericamente chiamata hardware qualunque parte di un sistema di elaborazione dei dati.

HOLD, TEMPO DI (Hold Time)
Il tempo che di solito deve rimanere stabile al termine del segnale di scrittura.

I

I.C.
Integred Circuit, cioè Circuito Integrato.

IN-CIRCUIT EMULATOR (ICE)
Emulatore nel circuito. Supporto per effettuare il debug; viene inserito nello zoccolo del microprocessore del sistema sotto esame. Questo fa sì che l'ICE possa acquistare il controllo completo del sistema. Caratteristiche tipiche sono le possibilità di inserire dei breakpoint, di eseguire passo passo un programma, di esaminare e modificare registri e memorie e di suddividere la memoria e gli I/O tra il sistema sotto esame e il sistema cui appartiene l'ICE.

INCREMENTARE (Increment)
Aggiungere il valore uno al contenuto di un registro o di una locazione di memoria.

INDICE, REGISTRO (Index Register)
Contiene l'informazione di indirizzo utilizzata per l'indirizzamento indicizzato.

INDIRIZZAMENTO DIRETTO (Direct Addressing)
Modo standard di indirizzamento caratterizzato dalla possibilità di accedere ad ogni punto della memoria principale in modo diretto. L'indirizzo viene specificato da una parte della istruzione.

INDIRIZZAMENTO IMMEDIATO (Immediate Addressing)
In questo modo di indirizzamento, l'istruzione contiene essa stessa il valore su cui si deve operare e non si hanno perciò riferimenti ad altri indirizzi.

INDIRIZZAMENTO INDICIZZATO (Indexed Addressing)
Modo di indirizzamento in cui l'indirizzo effettivo è ottenuto sommando uno spiazzamento (displacement) a un indirizzo base.

INDIRIZZAMENTO INDIRETTO (Indirect Addressing)
Indirizzare una locazione di memoria che contiene non il dato, ma l'indirizzo del dato.

INDIRIZZAMENTO, MODI DI (Addressing Modes)
Svariati metodi di specificare un indirizzo come parte di una istruzione.

INDIRIZZAMENTO RELATIVO (Relative Addressing)
Specificare un indirizzo dandone la distanza dall'indirizzo corrente: p.es. quattro byte indietro o tre byte avanti.

INDIRIZZI, BUS DEGLI (Address Bus)
Gruppo di fili (in genere 16) utilizzato per trasmettere indirizzi, di solito dal microprocessore a un dispositivo di memoria o di I/O.

INDIRIZZO (Address)
Numero che indica la posizione nella memoria di una parola. In genere gli indirizzi sono di sedici bit e possono quindi avere valori compresi tra 0 e 64 K.

INGRESSO /USCITA
Si veda INPUT/OUTPUT.

INIZIALIZZAZIONE (Initialization)
Portare il sistema in uno stato conosciuto.

INPUT/OUTPUT (I/O)
Linee o dispositivi utilizzati per scambiare informazioni con il mondo esterno al sistema.

INTERFACCIA (Interface)
Indica un confine, tra componenti, circuiti o sistemi adiacenti, che permette ai dispositivi di scambiare informazioni. Con tale nome si indica anche il circuito che abilita il microprocessore a comunicare con un dispositivo periferico.

INTERFERENZA ELETTROMAGNETICA (EMT) (Electromagnetic Interference)
Interferenza provocata da campi elettrici.

INTERRUZIONE (Interrupt)
Sospensione del programma normale, che il microprocessore sta eseguendo, al fine di gestire una improvvisa richiesta di servizio (interruzione). Il processore, dal programma che stava eseguendo, salta quindi alla routine di servizio dell'interruzione. Quando questa ultima è terminata, il controllo ritorna al programma interrotto.

INTERRUZIONE, MASCHERA DI (Interrupt Mask)
Registro che ha un bit di controllo per ogni interruzione. Utilizzato per disabilitare le varie interruzioni in modo selettivo.

INTERRUZIONE, ROUTINE DI SERVIZIO DELLA (Interrupt Service Routine)
Programma che viene eseguito quando si verifica una interruzione.

INTERRUZIONE VETTORIZZATA (Vectored Interrupt)
Quando in risposta al segnale di riconoscimento dell'interruzione viene presentato un numero di identificazione del dispositivo o un indirizzo effettivo di salto. Ogni interruzione può così essere automaticamente servita da una routine diversa.

INTERVAL TIMER
Dispositivo programmabile utilizzato per funzioni di temporizzazione, conteggio e ritardo. Di solito viene considerato e trattato come una periferica.

I/O CON MAPPA MEMORIA (Memory Mapped I/O)
Dispositivi di I/O l'accesso ai quali è effettuato utilizzando lo stesso gruppo di istruzioni e di segnali di controllo che sono usati, in un sistema, per i dispositivi di memoria. I dispositivi di I/O e quelli di memoria si spartiscono lo stesso spazio degli indirizzi separato.

ISTRUZIONE (Instruction)
Un comando all'interno di un programma. Le istruzioni possono essere aritmetiche o logiche, possono operare sui registri, sulla memoria o sui dispositivi di I/O, o possono specificare delle operazioni di controllo. Un programma è formato da una sequenza di istruzioni.

ISTRUZIONE, CICLO (Instruction Cycle)
Tutti gli stati macchina necessari all'esecuzione completa di una istruzione.

ISTRUZIONI, DECODER DELLE (Instruction Decoder)
Unità che interpreta le istruzioni del programma e genera, in corrispondenza, dei segnali di controllo per il resto del sistema.

ISTRUZIONI, INSIEME DELLE (Instruction Set)
Il gruppo di tutte le istruzioni che possono essere eseguite da un certo microprocessore. È fornito all'utente quale informazione fondamentale per assemblare un programma. È anche detto Set di Istruzioni.

ISTRUZIONI, REGISTRO (Instruction Register)
Registro interno al microprocessore che contiene il codice operativo dell'istruzione la cui esecuzione è in corso.

ITERATIVO (Iterative)
Processo o procedura che esegue ripetutamente una serie di operazioni finché non sia soddisfatta una qualche condizione. Di solito, in un programma, viene realizzato con un loop.

J

JUMP
Si veda: SALTO.

K

K
Simbolo che indica 1000 (103). Quando è riferito a bit o a parole, K = 1024 (2^{10}).

KERNEL
Si veda: NOCCIOLO.

L

LABEL
Etichetta. Nome assegnato ad una locazione di memoria. Quando si scrive un programma in linguaggio di assemblaggio, si assegnano delle label a quelle istruzioni o locazioni di memoria cui altre istruzioni dovranno fare riferimento. Quando il programma è poi convertito in codice macchina, ad ogni label verrà assegnato un valore effettivo di indirizzo.

LATCH
Dispositivo hardware che cattura e mantiene delle informazioni (p. es. un gruppo di flip-flop).

LED
Light Emitting Diode, cioè Diodo che Emette Luce. Dispositivo semiconduttore che, quando percorso da corrente, emette luce.

LIFO
Last-In-First-Out, cioè Ultimo a Entrare - Primo a Uscire. Lo stesso che STACK.

LINGUAGGIO AD ALTO LIVELLO (High-Level Language)
Linguaggio di programmazione orientato al problema, a differenza dei linguaggi di programmazione orientati alla macchina. Un linguaggio ad alto livello è più vicino alle esigenze del problema da trattare che al linguaggio della macchina su cui deve essere eseguito il programma.

LINGUAGGIO DI PROGRAMMAZIONE (Programming Language)
Linguaggio utilizzato per scrivere un programma. Si possono avere linguaggi macchina, di assemblaggio o ad alto livello.

LINGUAGGIO MACCHINA (Machine Language)
Linguaggio binario (spesso rappresentato in esadecimale) direttamente comprensibile per il processore. Tutti gli altri linguaggi di programmazione devono essere tradotti in codice binario prima che possano essere introdotti nel processore.

LOGIC PULSER
Si veda GENERATORE DI IMPULSI LOGICI.

LOGICA CABLATA (Hard-Wired Logic)
Si veda: LOGICA SPARSA.

LOGICA COMBINATORIA (Combinational Logic)
Organizzazione di un circuito per cui lo stato d'uscita è determinato solo dallo stato attuale degli ingressi.

LOGICA NEGATIVA (Negative Logic)
Lo stato logico falso è rappresentato dalla tensione più positiva del sistema, mentre lo stato logico vero è rappresentato dalla tensione più negativa del sistema. Nel caso della famiglia TTL, + 2,4 V (o più) corrispondono a 0, mentre 0, 4 V (o meno) corrispondono a 1.

LOGICA POSITIVA (Positive Logic)
Il livello vero è il livello di tensione più positivo del sistema.

LOGICA SEQUENZIALE (Sequential Logic)
Tipo di circuito in cui lo stato d'uscita è determinato dallo stato precedente e dagli ingressi attuali. (Si faccia il confronto con LOGICA COMBINATORIA).

LOGICA SPARSA (Random Logic)
Le soluzioni progettuali a Logica Cablata (o sparsa) richiedono il collegamento di numerosi circuiti integrati, che rappresentano i diversi elementi logici. Piuttosto che da un programma, la funzione del circuito è fissata dai blocchi funzionali e dai loro collegamenti.

LOOP
Anello. Parte di un programma che viene eseguita più volte.

LSB
Si veda BIT MENO SIGNIFICATIVO.

LSI
Large Scale Integration, cioè Integrazione a Larga Scala. Tecnologia grazie alla quale su di un singolo chip vengono fabbricati migliaia di dispositivi a semiconduttore.

LSTTL
Low power/Schottky TTL. Circuiti Integrati Digitali che utilizzano dei transistor Schottky per migliorare, rispetto ai dispositivi TTL standard, le caratteristiche di velocità/potenza.

M

MANTISSA (Mantissa)
Valore frazionario utilizzato come parte di un numero in virgola mobile. Ad esempio, la mantissa del numero $0{,}9873 \times 10^7$ è 0,9873.

MAPPA DI MEMORIA (Memory Map)
Mostra gli indirizzi assegnati ad ogni dispositivo del sistema.

MASCHERA (Mask)
Configurazione utilizzata per portare, in modo selettivo, alcuni bit di una parola a 0 o a 1. Viene di solito posta in AND o in OR con il dato.

MATRICE A PUNTI (Dot Matrix)
Metodo per formare dei caratteri utilizzando numerosi e minuscoli punti.

MEMORIA (Memory)
Parte di un calcolatore in cui le informazioni possono essere inserite, mantenute o recuperate. I termini inglesi «storage» e «memory» sono del tutto equivalenti. Le memorie digitali accettano e immagazzinano solo numeri binari. Tra i tipi di memorie più diffusi si hanno quelle a nuclei, a dischi, a nastro e a semiconduttore (che comprendono ROM e RAM).

MEMORIA DI MASSA (Mass Storage)
Memoria secondaria, più lenta, per grossi archivi. Di solito floppy disk o cassette.

MEMORIA DINAMICA (Dynamic Memory)
Dispositivi di memoria in cui i dati contenuti devono essere, per evitare una loro perdita, continuamente rinfrescati. Ogni bit è memorizzato, come una carica, all'interno di un solo condensatore MOS. A causa delle perdite di carica nei transistor, la memoria dinamica deve essere rinfrescata ogni 2 ms riscrivendone, in generale, l'intero contenuto; tale operazione non rallenta il sistema ma richiede della logica ulteriore per il rinfresco della memoria.

MEMORIA STATICA (Static Memory)
Dispositivi di memoria che non richiedono clock né rinfresco.

MEMORIA VOLATILE (Volatile Memory)
Dispositivi di memoria in cui i dati contenuti vengono alterati allo spegnimento dell'alimentazione. È possibile rendere non volatili le RAM fornendole di alimentatori o batterie di riserva (dette di back-up).

MICROCALCOLATORE (Microcomputer)
Sistema completo, comprendente CPU, memoria e interfacce di I/O.

MICROCODICE (Microcode)
Si veda: MICROPROGRAMMA.

MICROPROCESSORE (Microprocessor)
Unità centrale di processo realizzata in uno o due chip. Il processore è formato dalla unità aritmetico-logica, da una sezione di controllo e dai registri.

MICROPROGRAMMA (Microprogram)
Programma che definisce l'insieme delle istruzioni. Il microprogramma (detto anche microcodice) indica alla CPU che cosa deve fare al fine di eseguire ogni istruzione in linguaggio macchina. Entra anche, più del linguaggio macchina, nei dettagli e non è in genere accessibile all'utente.

MODEM
Modulatore - Demodulatore. Solitamente utilizzato per interfacciare un dispositivo digitale con una linea telefonica. Codifica e decodifica i bit seriali in frequenze.

MONITOR
Programma che controlla le operazioni di un sistema microcalcolatore e permette all'utente di eseguire programmi, esaminare ed alterare la memoria, ecc.

MOS
Metal Oxide Semiconductor, cioè Metallo Ossido Semiconduttore. Circuiti integrati realizzati con transistor ad effetto di campo. Inizialmente tutti i dispositivi MOS utilizzano infatti la tecnologia a gate di metallo; il termine è rimasto anche per descrivere i circuiti con gate di silicio.

MOTHERBOARD
Si veda: BACKPLANE.

MPU
Microprocessing Unit, cioè Unità di Microprocesso. Si veda MICROPROCESSORE.

MSB
Si veda: BIT PIÙ SIGNIFICATIVO.

MSI
Medium Scale Integration, cioè Integrazione a Media Scala. Tecnologia grazie alla quale su di un chip sono inserite funzioni che richiedono una dozzina o più di dispositivi elementari (gate).

MTBF
Mean Time Between Failures, cioè Tempo Medio tra Malfunzioni.

MTTR
Mean Time to Repair, cioè Tempo Medio per la Riparazione.

MULTIPLEX
Procedimento per trasmettere più di un segnale su di un singolo canale (collegamento). Nei sistemi a microprocessore la tecnica più diffusa è quella di avere un multiplex con divisione del tempo: una linea di segnale viene utilizzata per trasferire diverse informazioni in tempi diversi.

N

NIDIFICATO
Detto in modo più aderente alla funzione: innestato. Una subroutine che è chiamata da un'altra subroutine o un loop all'interno di un loop più grande sono detti nidificati.

NOCCIOLO (Kernel)
Il circuito minimo che permette al microprocessore di funzionare. Di solito è formato dal microprocessore, dal circuito di clock, dalle linee di controllo interruzione e DMA e dalla alimentazione.

NODO (Node)
Qualunque linea di segnale collegata a due o più elementi di un circuito. Tutte le uscite e gli ingressi logici elettricamente collegati appartengono allo stesso nodo.

NUCLEO (Core)
Piccoli anelli magnetici toroidali di ferrite che vengono utilizzati per immagazzinare un bit d'informazione. Possono essere infilati su fili in modo da formare grosse strutture di memoria. Il vantaggio principale delle memorie a nuclei è quello di essere non volatili.

NUMBER CRUNCHING
Macinare Numeri, effettuare complesse operazioni numeriche.

O

OPCODE
Si veda: CODICE OPERATIVO.

ORDINE ALTO (High-Order)
I bit più significativi di una parola. Tipicamente i bit da 8 a 15 di una parola a 16 bit.

ORDINE BASSO (Low Order)
Fa riferimento al peso, cioè al significato, assegnato alle cifre di un numero. Nel numero 123456, la cifra di ordine più basso è 6. Nella parola binaria 11100101 i tre bit di ordine basso sono 101.

OTTALE (Octal)
Sistema numerico in base 8. Spesso utilizzato per rappresentare numeri binari poiché ogni cifra ottale corrisponde pari pari a tre cifre binarie.

OVERFLOW
Trabocco, supplemento. Si verifica quando un'operazione aritmetica dà come risultato un valore più grande della capacità del registro. Se un'operazione ha generato un overflow, viene portato a uno nel registro dei flag un bit di stato dell'overflow (il bit di carry).

P

PAGINA (Page)
Solitamente un blocco di 256 indirizzi. Gli otto bit più bassi, di una parola d'indirizzo, specificano la locazione all'interno della pagina, mentre gli otto bit più alti specificano la pagina.

PARAMETRO (Parameter)
Valore che viene passato da una routine ad un'altra, attraverso un registro o attraverso una locazione di memoria.

PARITA' (Parity)
Numero di 1 contenuti in una parola; può essere pari o dispari. Quando si usa la parità, ci si serve di un bit supplementare per far sì che il numero di 1 nella parola (compreso il bit di parità) sia pari (e si parla allora di parità pari) o dispari (parità dispari). La parità è una delle tecniche più semplici per rivelare gli errori e serve a rivelare errori sui singoli bit.

PARLATORE (Talker)
Dispositivo che può introdurre dei dati sul bus dei dati. Una ROM è un «parlatore».

PAROLA (Word)
Insieme di caratteri che occupa una locazione di memoria e che i circuiti del calcolatore trattano come una entità unica. Di solito l'unità di controllo tratta le parole come istruzioni, mentre l'unità aritmetica le considera delle quantità.

PATCH
Pezza. Parte di codice inserita all'interno di una routine per correggere un errore o per alterare la routine stessa. Di solito non è inserita all'interno della sequenza effettiva della routine che deve essere corretta, ma è posta da qualche altra parte. Sono perciò introdotti un salto alla «pezza» e un ritorno alla routine.

PC
Program Counter, cioè Contatore di Programma. In inglese sta anche per Printed Circuit, cioè Circuito Composto.

PCB
Printed Circuit Board, cioè Scheda a Circuito Stampato.

PERIFERICA (Peripheral)
Qualunque dispositivo di interfaccia collegato ad un calcolatore. È detto anche di memorie di massa o di dispositivi di comunicazione collegati a un calcolatore.

POLLING
Un metodo utilizzato per identificare quale dispositivo ha generato una richiesta d'interruzione. La CPU deve effettuare un «polling» (una interrogazione), cioè una lettura, verso tutti i dispositivi, al fine di stabilire quale di essi ha causato una interruzione.

POP
Operazione di lettura di una parola dallo stack; ha lo stesso significato di «pull» (tirare, estrarre).

PORTA (Port)
Punto in cui i dispositivi di I/O sono collegati al calcolatore.

PORTA DI INGRESSO (Input Port)
Circuito che collega come ingressi al sistema a microprocessore i segnali provenienti da dispositivi esterni.

PORTA DI USCITA (Output Port)
Circuito che permette al sistema a microprocessore di inviare dei segnali verso altri dispositivi.

POWER-UP RESET
Si veda RESET ALL'ACCENSIONE.

PRIORITA' (Priority)
Numero, assegnato ad un evento o a un dispositivo, che definisce l'ordine secondo cui verrà servito nel caso si verificasse nello stesso istante più di una richiesta.

PROCESSORE (Processor)
Si veda: MICROPROCESSORE.

PROGRAM COUNTER (PC)
Contatore di programma. Registro interno alla CPU che contiene l'indirizzo del prossimo byte di programma che deve essere letto. I salti e le deviazioni dalla sequenza di istruzioni richiedono che il program counter venga caricato con l'indirizzo del salto. In tutti gli altri casi il PC viene incrementato dopo ogni byte letto.

PROGRAMMA (Program)
Procedura per risolvere un problema, codificata in una forma tale da poter essere utilizzata da un calcolatore. Spesso viene anche indicato con il termine software.

PROGRAMMA OGGETTO (Object Program)
Risultato finale di un programma scritto in linguaggio sorgente (d'assemblaggio o ad alto livello) al termine dell'operazione di traduzione in linguaggio macchina.

PROGRAMMATO DA MASCHERA (Mask Programmed)
Un circuito integrato che è programmato attraverso l'utilizzo di una particolare maschera fotografica nel processo di costruzione dello stesso.

PROM
Programmable Read-Only Memory, cioè Memoria Programmabile a Sola Lettura. Memoria a circuito integrato che viene prodotta con un contenuto di tutti 0 o tutti 1 e in cui le diverse configurazioni possono essere scritte utilizzando uno speciale programmatore hardware.

PROTOCOLLO (Protocol)
Insieme di regole per gli scambi di informazioni.

PSEUDO-ISTRUZIONE (Pseudo-Instruction)
Istruzione che viene utilizzata all'interno di un programma in linguaggio di assemblaggio, ma che è in effetti un'istruzione per l'assembler. Le pseudo-istruzioni non hanno una diretta corrispondenza con il lingaggio macchina.

PULL-UP RESISTOR
Si veda: RICHIAMO, RESISTENZE DI.

PUNTATORE DELLO STACK (Stack Pointer)
Contiene l'indirizzo della cima dello stack. In genere il puntatore dello stack viene decrementato subito dopo la memorizzazione di un byte di informazioni nello stack. Viceversa, il puntatore dello stack viene incrementato appena prima del prelievo di un byte di informazioni dallo stack.

PUSH
L'operazione di introdurre una parola nello stack.

PUSH-DOWN STACK
Si veda STACK.

R

RADICE (Radix)
Il numero totale di caratteri diversi, cioè di numeri, che sono utilizzati in un sistema di numerazione. È detta anche base.

RAM
Random Access Memory, cioè Memoria ad Accesso Casuale. Di solito vengono così denominate le memorie lettura/scrittura a semiconduttore. In effetti, in senso stretto, anche le ROM sono delle RAM (si veda anche ACCESSO CASUALE).

REAZIONE (Feedback)
Informazioni che, prese da una o più uscite, vengono utilizzate come ingressi in un anello di controllo.

REFRESH
Si veda: RINFRESCO.

REGISTRO (Register)
Una singola parola di memoria. È più immediato accedere ai registri interni alla CPU che alle locazioni esterne. All'esterno della CPU i registri sono solo dei gruppi di flip-flop.

RESET ALL'ACCENSIONE (Power-Up Reset)
Processo di inizializzazione per cui gli elementi di memoria presenti in un sistema, ogni volta che quest'ultimo viene acceso, vengono portati a delle ben definite condizioni.

RICERCA BINARIA (Binary Search)
Tecnica secondo la quale ad ogni iterazione l'intervallo di ricerca viene diviso per due.

RICERCA GUASTI (Troubleshoot)
Ricerca la causa di una malfunzione o di un comportamento errato di un programma al fine di eliminare il guasto o l'errore.

RICERCA GUASTI, ALBERO DI (Troubleshooting Tree)
Diagramma di flusso che contiene le prove e le misure che devono essere effettuate al fine di diagnosticare e localizzare i guasti presenti in un prodotto.

RICHIAMO, RESISTENZA DI (Pull-Up Resister)
Utilizzata come generatore di corrente per dispositivi logici open-collector e a tre stati o come terminazione per ingressi non utilizzati. Quando nessun altro dispositivo sta pilotando la linea, porta alto il livello di tensione.

RIMBALZO (Bounce)
Oscillazioni e rumori che si verificano quando un interruttore meccanico si apre o si chiude. Si veda: ANTIRIMBALZO.

RINFRESCO (Refresh)
Il procedimento che, in una memoria dinamica, ripristina la carica. La logica di rinfresco deve riscrivere periodicamente (tipicamente ogni 2 ms) il contenuto di tutta la RAM, effettuando così l'operazione detta «rinfrescare la memoria». (Si veda MEMORIE DINAMICHE).

RIPORTO, FLAG DI
Si veda CARRY, FLAG DI.

RIPRISTINARE (Restore)
Riportare un registro o un'altra parola di un calcolatore al suo valore iniziale o al suo valore prefissato.

RITARDO DI PROPAGAZIONE (Propagation Delay)
Tempo che un segnale impiega per propagarsi attraverso un dispositivo.

RITORNO (Return)
Meccanismo che serve ad effettuare un ritorno, esattamente come nel senso comune. In particolare fa riferimento alla istruzione, posta alla conclusione di una subroutine, che fa sì che il controllo ritorni al punto opportuno della routine principale.

ROM
Read-Only Memory, cioè Memoria a Sola Lettura. Memoria programmata in modo permanente. Le ROM programmate da maschera sono programmate dal fabbricante dei chip. Le PROM (ROM programmabili) possono essere programmate dall'utente. Le EPROM (PROM cancellabili) possono essere cancellate con luce ultravioletta.

S

SALTO
Istruzione che altera la sequenza di esecuzione.

SCANSIONE (Scanning)
Procedimento di accesso sequenziale alle singole linee di segnale all'interno di un gruppo di linee.

SCHMITT TRIGGER
Circuito con isteasi utilizzato per segnali d'ingresso su cui possa essere presente del rumore o che abbiano transizioni lente.

SCRATCHPAD
Memoria che contiene dei risultati parziali necessari per costruire i risultati finali.

SCRIVERE (Write)
Trasferire delle informazioni, di solito dalla memoria principale ad un dispositivo d'uscita; registrare dei dati in un registro, una locazione o un qualunque altro dispositivo di memoria.

SERIALE (Serial)
Trasmettere i bit di un dato uno per volta lungo un solo filo, anziché utilizzare un filo per ogni bit.

SET-UP, TEMPO DI (Set-Up Time)
Tempo che un dato deve essere valido prima di un segnale di scrittura.

SHIFT
Scorrere. Muovere a sinistra o a destra, di una unità di informazione, i caratteri. Per un numero binario ciò è equivalente a moltiplicare o dividere per due ad ogni shift.

SIGNATURE
Si veda FIRMA.

SIMULATORE (Simulator)
Programma speciale che simula l'operatività del microprocessore. È progettato in modo da eseguire dei programmi in linguaggio macchina su di una macchina diversa da quella per cui tali programmi sono stati scritti. Consente l'utilizzo di programmi, scritti per un certo microprocessore, su di un sistema che fa uso di un altro processore.

SINGLE-STEP
Passo singolo. Processo per cui un programma viene eseguito un ciclo macchina o una istruzione alla volta.

SINK CURRENT
La corrente che può entrare in un dispositivo.

SOFTWARE
Si veda: PROGRAMMI

SONDA LOGICA (Logic Probe)
Strumento portatile di ricerca guasti che rivela gli stati logici e l'attività dei nodi digitali.

SOMMATORE (Adder)
Dispositivo che in uscita dà la somma di due o più numeri presenti ai suoi ingressi.

SOURCE CURRENT
Corrente che può uscire da un dispositivo.

SSI
Small Scale Integration, cioè Piccola Scala d'Integrazione. Tecnologia di complessità inferiore a quella a media scala d'integrazione. Di solito significa che in un circuito integrato sono inserite funzioni che richiedono meno di dieci dispositivi elementari (gate).

STACK
Catasta, Pila. Blocco di locazioni di memoria contigue cui è possibile accedere in modo LIFO. (Si veda la descrizione data a tale voce). Nella maggior parte dei processori lo stack può essere un qualunque blocco di locazioni contigue poste nella memoria a lettura/scrittura.

STATO (Status)
Condizione attuale di un dispositivo. Di solito viene indicata da flip-flop di flag o da registri particolari. Si veda FLAG.

STORAGE
Si veda: MEMORIA.

STRESS TESTING
Prova di sforzo. Sottoporre dei dispositivi elettrici a degli sforzi meccanici, elettrici o termici in modo da modificare il loro comportamento e permettere l'osservazione di guasti intermittenti.

SUBROUTINE
Parte «chiusa» di un programma che effettua un compito ben definito. Nello stesso programma può essere usata in più punti.

SVILUPPO, SISTEMA DI (Development System)
Sistema microcalcolatore con tutte le possibilità richieste per sviluppare l'hardware e il software relativi ad un certo microprocessore. È in genere formato da un sistema microcalcolatore, un display a CRT, una stampante, una memoria di massa (di solito due floppy-disk), un programmatore di PROM e un in-circuit emulator (emulatore nel circuito).

T

TABELLA (Table)
Raccolta di dati ordinati in modo da essere facilmente accessibili, spesso memorizzati in locazioni di memoria contigue.

TABLE LOOK-UP
Prendere un valore da una tabella di valori memorizzata nel calcolatore.

TASTIERA (Keyboard)
Insieme di pulsanti utilizzati per introdurre delle informazioni in un sistema.

TESTER DI SCHEDE (Board Tester)
Macchina programmata in modo da stimolare automaticamente i circuiti di una scheda a circuito stampato verificandone le risposte. Possono essere rilevati e diagnosticati i guasti elettrici in modo da facilitarne la riparazione.

TRE STATI (Three-State)
Dispositivo logico la cui uscita, oltre che poter essere nei soliti stati alto e basso, può anche essere posta in uno stato ad alta impedenza. Grazie a tali caratteristiche è possibile porre sullo stesso nodo logico più di una uscita di dispositivi. Il modo di operare a tre stati è una esigenza essenziale per i dispositivi che sono utilizzati sui bus dei dati dei microprocessori. Ha lo stesso significato di tri-state (marchio registrato).

TROUBLESHOOT
Si veda RICERCA GUASTI.

TROUGHPUT
Velocità con cui problemi o parti di problemi vengono risolti. È un valore che può variare da un'applicazione ad un'altra.

TTL
Transistor Transistor Logic. Famiglia di circuiti integrati digitali che hanno in ingresso e in uscita dei transistor bipolari.

TTY
Teletype, Telescrivente.

U

UART
Universal Asynchronous Receiver Transmitter, cioè Ricevitore e Trasmettitore Asincrono Universale. Un convertitore da seriale a parallelo e da parallelo a seriale.

μC
Microcalcolatore.

UNIDIREZIONALE (Unidirectional)
Filo o gruppo di fili che portano i dati in una sola direzione. Ogni dispositivo collegato a un bus unidirezionale può essere un trasmettitore o un ricevitore, ma non entrambi.

UNITA' ARITMETICO-LOGICA, ALU (Arithmetic and Logic Unit)
Una delle tre parti fondamentali di un microprocessore. Le altre due sono i registri e la parte di controllo. La ALU effettua diversi tipi di somme, sottrazioni ed operazioni logiche, come l'operazione di AND tra il contenuto di due registri o l'operazione di mascheratura del contenuto di un registro.

UNITA' CENTRALE DI PROCESSO
Si veda: CPU.

μP
Microprocessore.

V

VIRGOLA FISSA, RAPPRESENTAZIONE A (Fixed-Point Representation)
Rappresentazione numerica in cui si assume che la virgola decimale si trovi in una posizione fissa.

VIRGOLA MOBILE, RAPPRESENTAZIONE IN (Floating Point Representation)
Tecnica utilizzata per rappresentare un grande intervallo di numeri; fa uso di una mantissa e di un esponente. La precisione della rappresentazione è legata al numero di bit di cui è formata la mantissa. Si vedano MANTISSA ed ESPONENTE.

VLSI
Very Large Scale Integration, cioè Integrazione a Larghissima Scala. Tecnologia grazie alla quale sullo stesso chip vengono fabbricati centinaia di migliaia di dispositivi semiconduttori.

W

WALKING-ONE
Test delle memorie in cui in tutte le locazioni di memoria caricate con 0, viene fatto muovere un bit a 1. Simmetrico di questo è il test «walking-zero».

BIBLIOGRAFIA

BIBLIOGRAFIA

LIBRI

Blakeslee, Thomas R. *Digital Design with Standard MSI and LSI,* 2nd Ed. Somerset, New Jersey: Wiley-Interscience, 1979. Descrive le tecniche di progettazione digitale facendo uso di logica standard e di microprocessori.

Hilburn, John L. e Julich, Paul M. *Microcomputers/Microprocessors: Hardware, Software, and Applications.* Englewood Cliffs, New Jersey: Prentice-Hall, 1976. Un libro generale sui principi dei microcalcolatori.

Klingman, Edwin E. *Microprocessor Systems Design.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977. Un testo avanzato sulla progettazione dei sistemi a microprocessore.

Lesea, Ausin, e Zaks, Rodnay. *Microprocessor Interfacing Techniques.* Berkeley: Sybex, Inc., 1977. Descrive numerose periferiche molto utilizzate nei sistemi a microprocessore e i circuiti da impiegare per interfacciarle a tali sistemi.

Leventhal, Lance A. *8080A/8085 Assembly Language Programming.* Berkeley: Osborne & Associates, Inc., 1978. Descrive l'insieme di istruzioni dell'8080/8085 e riporta numerosi programmi esemplificativi.

Osborne, Adam. *An Introduction to Microcomputers: Volume 1-Basic Concepts.* Berkeley: Osborne & Associates, Inc., 1976. Un testo generale di introduzione ai concetti dei microprocessori.

Osborne, Adam. *An Introduction to Microcomputers: Volume II-Some Real Products.* Berkeley: Osborne & Associates, Inc., 1978. Descrive numerosi microprocessori e chip di supporto disponibili.

Peatman, John B. *Microcomputer-Based Design.* New York: McGraw Hill, 1977. Descrive le tecniche di progettazione per strumenti basati su microprocessori.

Yourdon, Edward. *Techniques of Program Structure and Design.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975. Presenta le tecniche per scrivere correttamente dei programmi strutturati.

Zaks, Rodnay. *Microprocessors from Chips to Systems.* Berkeley: Sybex Inc., 1977. Un'introduzione ai microprocessori con esempi pratici.

RIVISTE

«Microcomputer System Reference Issue». *EDN*. November 20, 1978 (Pubblicato ogni anno). Descrive tutti i microprocessori disponibili e i loro chip di supporto. «Microprocessor Data Manual.» *Electronic Design*. October 11, 1978. (Pubblicato ogni anno). Descrive tutti i microprocessori disponibili.

Beckwith, John F. «Current Tracer: A New Way to Find Low Impedance Logic-Circuit Faults». *Hewlett-Packard Journal,* December 1976, pp. 2-8.

PUBBLICAZIONI DEI PRODUTTORI

MCS-85 Microcomputer Systems User's Manual. Santa Clara, California: Intel Corp., 1978. Manuale di riferimento hardware dell'8085.

MCS-80 Microcomputer Systems User's Manual. Santa Clara, California: Intel Corp., 1977. Manuale di riferimento hardware dell'8080.

8080/8085 Assembly Language Programming Manual. Santa Clara, California: Intel Corp., 1978. Manuale di riferimento software dell'8080 e dell'8085.

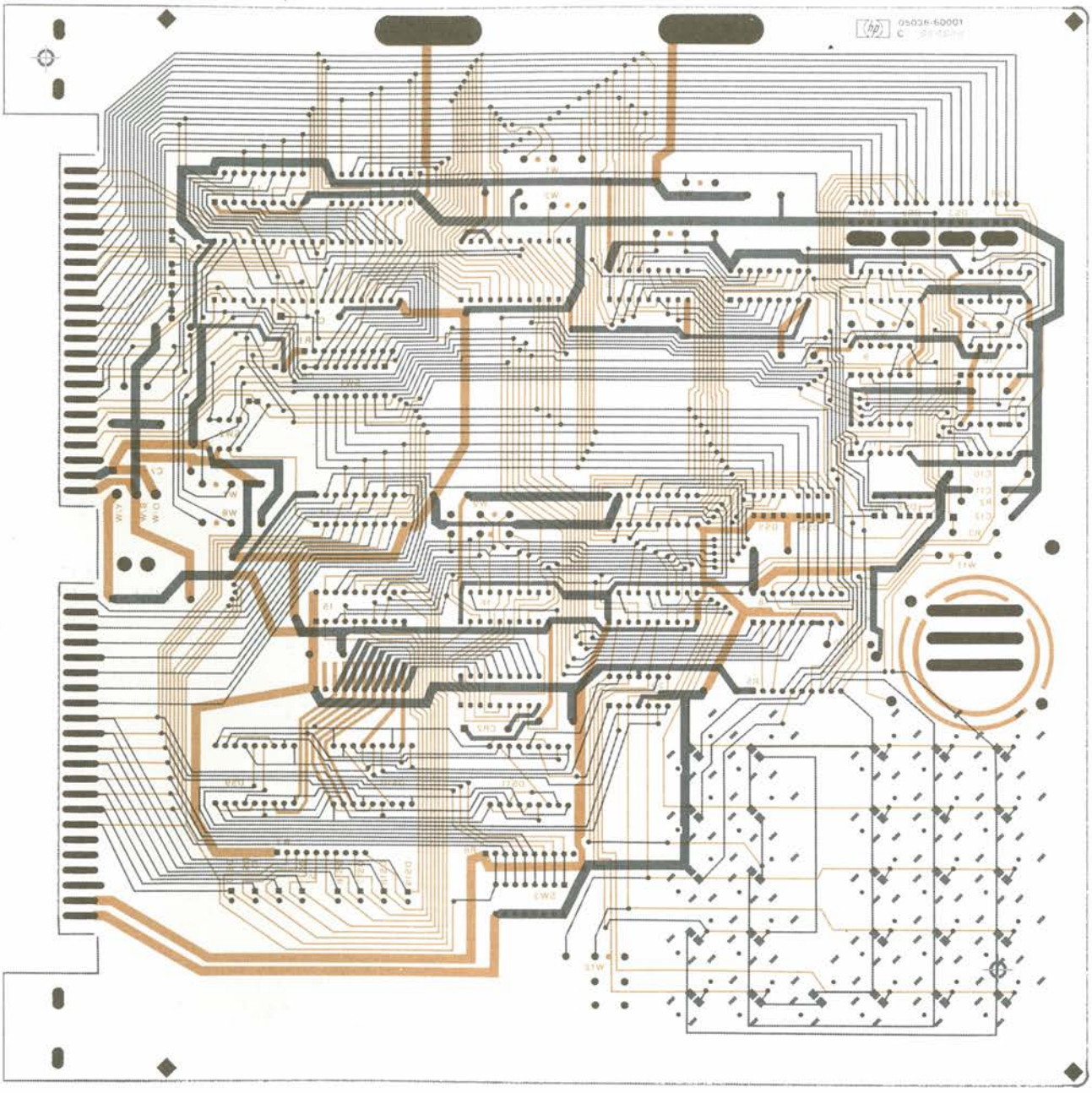
M6800 Microprocessor Applications Manual. Phoenix: Motorola Semiconductor Products, Inc., 1975. Una descrizione dell'hardware relativo al 6800 e un esempio di un progetto completo.

NOTE APPPLICATIVE

A Designer's Guide to Signature Analysis. Hewlett-Packard Application Note 222, 1977.

Techniques of Digital Troubleshooting. Hewlett-Packard Application Note 163-1, 1977.

DIAGRAMMI HARDWARE DI RIFERIMENTO



Schema elettrico del Microprocessor Lab 5036A

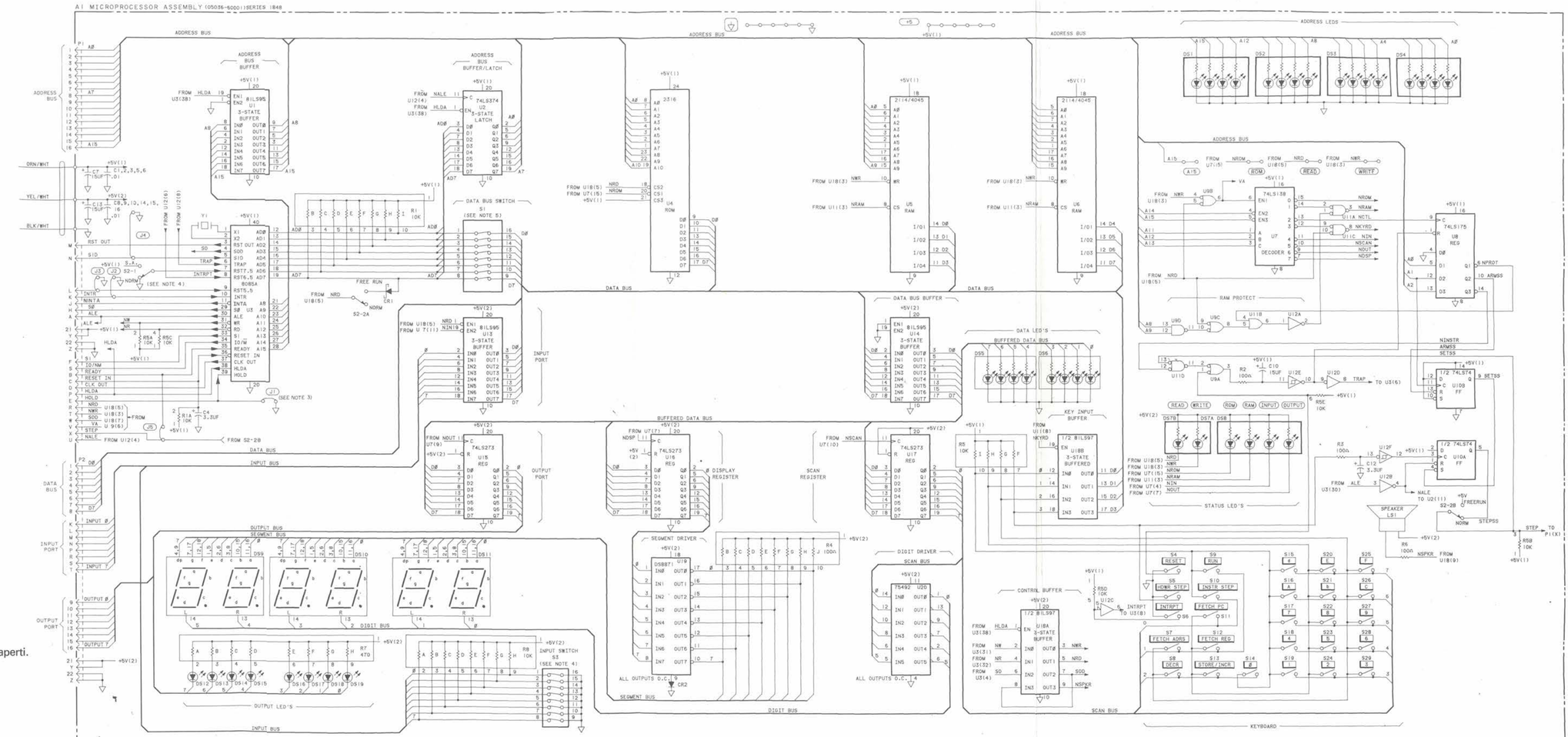
SCHEMA ELETTRICO DEL MICROPROCESSOR LAB 5036A
(vedi pagina 451)

SCHEMA A BLOCCHI DEL MICROPROCESSOR LAB 5036A
(vedi pagina 452)

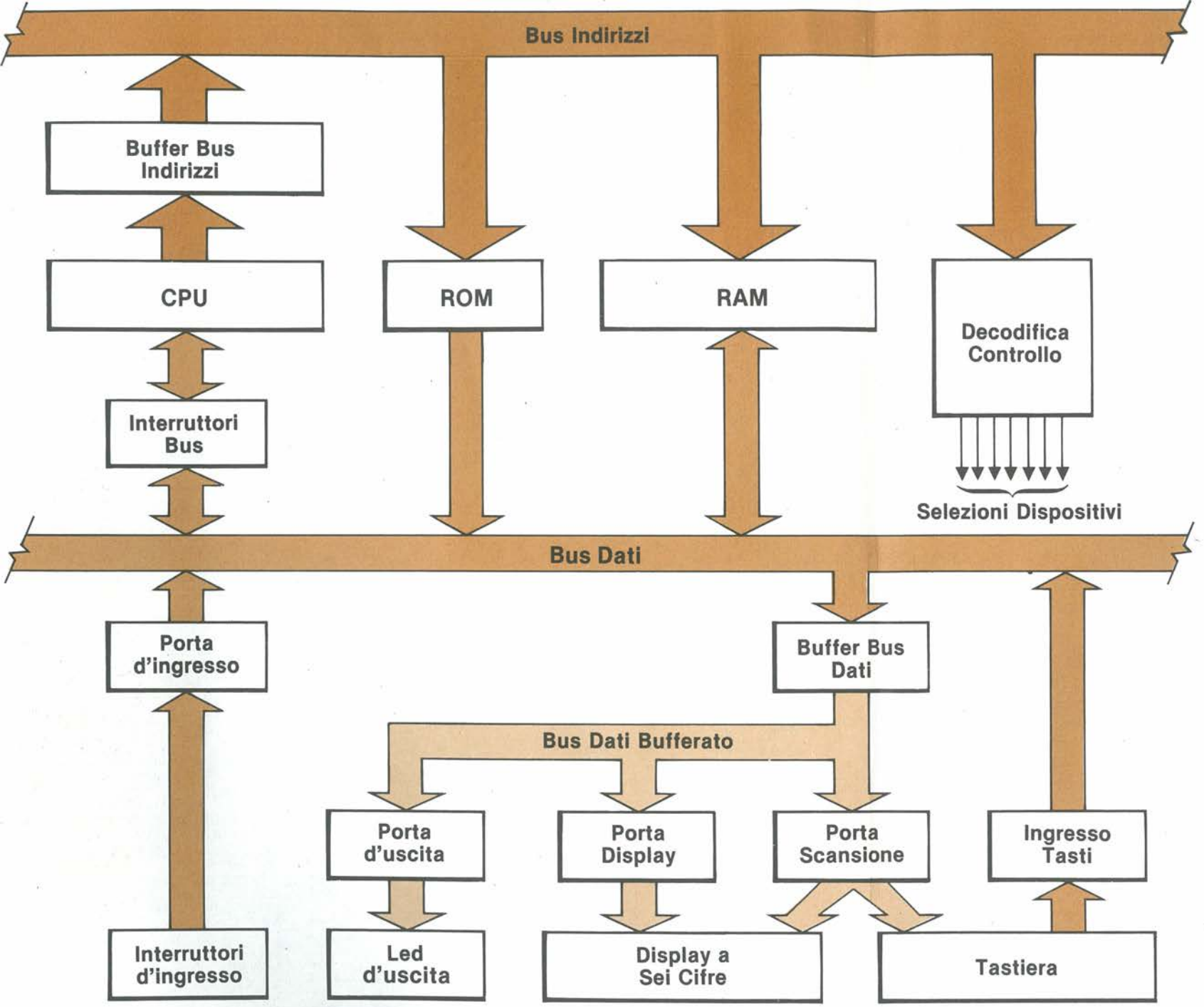
TABELLA DEI CIRCUITI INTEGRATI		
RIFERIMENTO	NUMERO CODICE HP	SIGLA GENERALE
U1, U13, U14	1820-1794	81LS95
U2	1820-1997	74LS374
U3	1820-2074	8085A
U4	1818-0773	1818-0773
U5, U6	1818-0438	2114
U7	1820-1216	74LS138
U8	1820-1195	74LS175
U9	1820-1197	74LS00
U10	1820-1112	74LS74
U11	1820-1208	74LS32
U12	1820-1416	74LS14
U15, U16, U17	1820-1730	74LS273
U18	1820-1759	87LS97
U19	1820-2138	8871
U20	1820-1231	75492

NOTE

1. All'interno di questo assemblaggio i riferimenti sono abbreviati. Per una descrizione completa aggiungete all'abbreviazione il numero d'assemblaggio.
2. Se non altrimenti indicato: le resistenze sono date in OHM, i condensatori in microfarad
3. Le sottili piste di collegamento dei ponticelli J1, J2, J3 e J4 devono essere tagliate se tali linee sono utilizzate con periferiche.
4. Gli interruttori S3 sono mostrati chiusi (0).
5. Gli interruttori S1 sono mostrati chiusi (posizione normale). Per il modo free-run tutti gli interruttori S1 devono invece essere aperti.
6. Le linee dei segnali attivi bassi sono indicate con un prefisso N (p. es.: NRAM = $\overline{\text{RAM}}$).



Schema piste della scheda 5036A



Schema a blocchi del Microprocessor Lab 5036A

Riassunto delle funzioni

RESET — Forza il μLab nel modo «attesa comando». Utilizzato per uscire dai programmi utente. Se premuto quando si è nel modo single-step hardware, viene visualizzato l'indirizzo della istruzione seguente. Non altera né il contenuto della memoria né quello dei registri (si veda la pag. 17).

HDWR STEP — Porta il μLab nel modo single-step hardware. All'interno di questo modo, ogni volta che viene premuto il tasto HDWR STEP tra il microprocessore e il resto del sistema viene trasferito un byte d'informazione. Con il sistema fermo, è possibile osservare direttamente sui LED binari dei bus e dello stato, presenti nel μLab, i trasferimenti dei dati e delle istruzioni (si veda la pag. 56).

INTRPT — Interrompe il μLab (se questa interruzione è stata abilitata). Utilizzato soprattutto per mostrare l'uso delle istruzioni (si vedano le pagg. 77 e 80).

FETCH ADRS — Specifica che volete introdurre un indirizzo. Il μLab si aspetta perciò che venga inserito un indirizzo di quattro cifre. Utilizzato per esaminare la memoria e per specificare la locazione di partenza di un programma (si veda la pag. 18).

DECR — Decrementa l'indirizzo visualizzato. Utilizzato per esaminare precedenti locazioni di memoria. Non altera il contenuto della memoria. Utilizzato anche per esaminare i registri (si veda la pag. 52).

RUN — Fa partire l'esecuzione del programma, il cui indirizzo di partenza è visualizzato. È il modo normale per far partire un programma (si veda la pag. 57).

INSTR STEP — Fa sì che l'istruzione visualizzata venga eseguita e che sia visualizzata l'istruzione seguente. Utilizzato per analizzare e collaudare i programmi (si veda la pag. 55).

FETCH P C — Visualizza come indirizzo l'ultimo valore del contatore di programma utente. Utilizzato soprattutto per ritornare ad un programma dopo un breakpoint. Utilizzato anche in single-step per ritornare ad un programma dopo che si siano esaminati registri e memoria (si vedano le pagg. 66 e 70).

FETCH REG — Porta il μLab nel modo «registro». Utilizzando i tasti STORE/INCR e DECR, è possibile esaminare e modificare i diversi registri (si vedano le pagg. 66, 70 e 171).

STORE /INCR — Memorizza il dato visualizzato all'indirizzo visualizzato ed incrementa quindi l'indirizzo. Utilizzato per introdurre dei dati e per esaminare locazioni di memoria una dopo l'altra. È anche utilizzato per esaminare ed alterare i registri. (si veda la pag. 51).

0 — F — Utilizzati per introdurre dati ed indirizzi esadecimali (si veda la pag. 51).

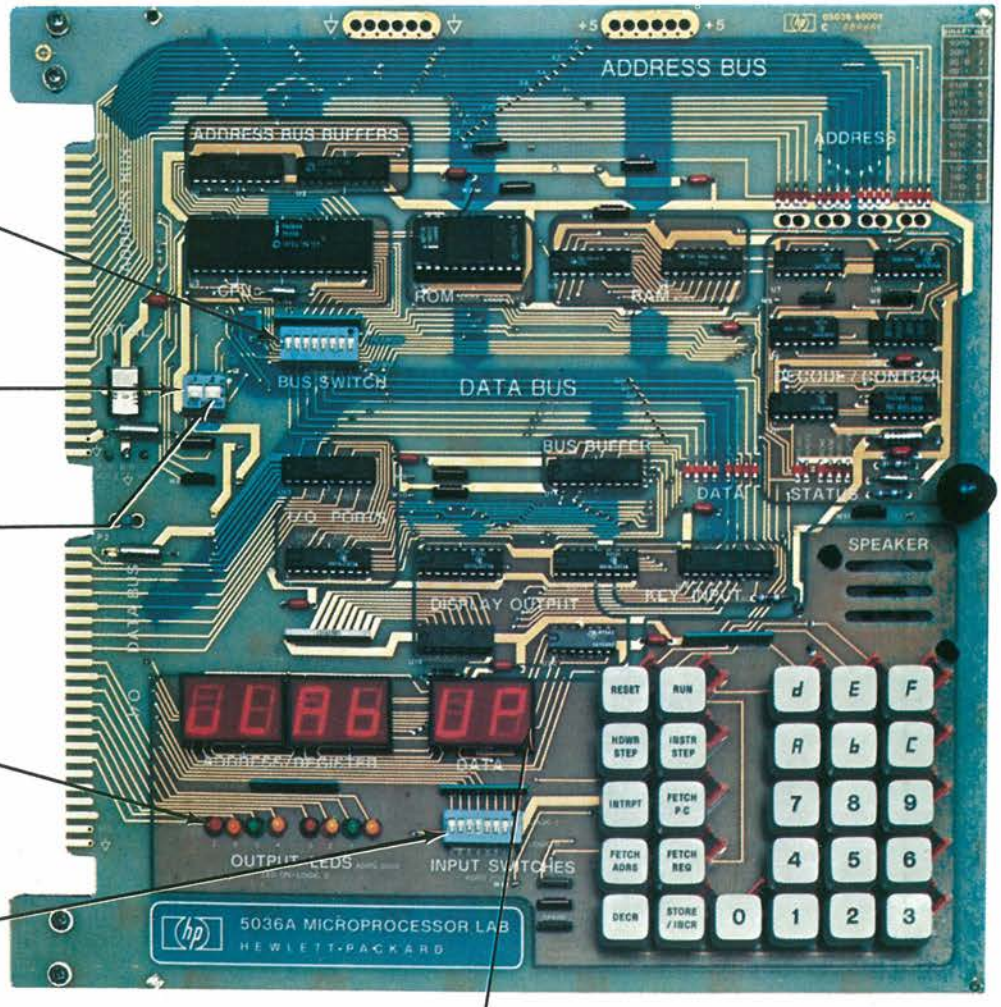
Data bus switches, interruttori del bus dei dati. Sono tutti verso l'alto per il modo free-run, tutti verso il basso in tutti gli altri casi (si veda la pag. 274).

Interruttore S.A. Per iniziare il modo di test Analisi della Firma (Signature Analysis, S.A.) muovetelo verso l'alto e poi verso il basso (si veda la pag. 283).

Interruttore di free-run. Verso l'alto per il modo free-run (si veda la pag. 274).

Output LEDS, cioè LED d'uscita, indirizzo 3000 (si veda la pag. 59).

Input switches, cioè Interruttori d'ingresso, indirizzo 2000 (si veda la pag. 59).



Nota: se è acceso il punto decimale, il μLab si trova nel modo ingresso dati e sono perciò disabilitati i tasti RUN, INSTR STEP e HDWR STEP (pag. 57).

Messaggi d'errore

Errori hardware:

«IC4», «1C5», «IC6»: indicano un problema hardware nel circuito Integrato (IC) visualizzato, nei circuiti che sono utilizzati con tale IC, o un ponticello di guasto mal posto.

Errori software:

«SP Er»: si ha un messaggio di errore del puntatore dello stack quando il vostro programma cerca di eseguire la sequenza di inizializzazione all'accensione posta all'indirizzo 0000. Questo errore si verifica di solito quando il programma esegue più ritorni che chiamate. Questo tipo di errore è molto comune all'interno dei programmi. Se il programma monitor non avesse rilevato questa condizione, il μLab avrebbe azzerato tutta la memoria con conseguente perdita del programma.

BIP dopo la pressione di STORE / INCR: vuol dire che il nuovo dato visualizzato non ha potuto essere inserito correttamente all'indirizzo specificato. Questo errore si verifica di solito quando si cerca di scrivere un dato in una locazione ROM (0000-07FF) o in un indirizzo esterno alla RAM (>0BFF). Si verifica anche quando si cerca di memorizzare un dato in una locazione di una RAM guasta.

BIP dopo la pressione di RUN: vuol dire che il vostro programma ha eseguito un'istruzione RST7 o, più probabilmente, non sta girando in un loop chiuso. Se non termina con un salto all'indietro in qualche punto all'interno del loop, il programma continuerà ad eseguire delle istruzioni in memoria (le istruzioni 00, NOP, memorizzate durante l'inizializzazione), finché all'indirizzo 0AEF verrà incontrata la istruzione RST7.A questo punto il μLab emette un bip, carica 0800 nel PC e ritorna al programma monitor.

All'interno la descrizione dei tasti

edizione
in lingua
italiana

PRACTICAL MICROPROCESSORS

JACKSON ITALIANA
EDITRICE